ANALISA DAN KONFIGURASI
NETWORK INTRUSION PREVENTION SYSTEM (NIPS)
PADA LINUX UBUNTU 10.04 LTS

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat Untuk
Memperoleh Gelar Sarjana Teknik Pada
Jurusan Teknik Informatika

oleh :

DISKHAMS MAULIDI MYDZA
10451025514

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI SULTAN SYARIF KASIM RIAU
PEKANBARU
2011

ANALISA DAN KONFIGURASI *NETWORK INTRUSION PREVENTION SYSTEM* (NIPS) PADA LINUX UBUNTU 10.04 LTS

DISKHAMS MAULIDI MYDZA

10451025514

Tanggal Sidang : 05 Juli 2011

Periode Wisuda : Oktober 2011

Jurusan Teknik Informatika

Fakultas Sains dan Teknologi

Universitas Islam Negeri Sultan Syarif Kasim Riau

ABSTRAK

Snort merupakan salah satu sistem pendeteksi penyusupan (*Intrusion Detection System*/IDS) yang *open source* dan banyak digunakan oleh *administrator* jaringan sebagai sistem untuk memonitor jaringan serta sebagai pendeteksi adanya serangan penyusupan pada pada jaringan. Fungsi Snort sebagai sistem pendeteksi penyusupan dapat dikembangkan menjadi sebuah sistem pencegah penyusupan (*Intrusion Prevention System*/IPS) dengan bantuan *firewall* (IPTables) dan Snortsam.

Snortsam merupakan sebuah *plugin* yang membuat *rule* Snort mampu memerintahkan *firewall* (IPTables) untuk mem-*block* paket data yang dideteksi sebagai penyusupan oleh Snort. Snort mengidentifikasi paket data tersebut sebagai sebuah penyusupan karena pola paket data tersebut sama dengan pola *rule* Snort yang mendefinisikan sebagai sebuah penyusupan. *Log* dari pendeteksian penyusupan tersebut disimpan sebagai *alert*.

Dalam tugas akhir ini, penulis akan mengkonfigurasi sistem pencegah penyusupan (*Intrusion Prevention System*/IPS) dengan menggabungkan Snort, Snortsam dan *firewall* (IPTables) pada sistem operasi Ubuntu. Alasan pemilihan Ubuntu sebagai sistem operasi pada konfigurasi sistem pencegah penyusupan ini karena Ubuntu merupakan sistem operasi Linux yang mudah digunakan dan dikembangkan sesuai dari keinginan penggunanya.

Kata Kunci: *firewall (IPTables), Intrusion Detection System, Intrusion Prevention System,* Linux*,* Snort, Snortsam*,* Snort *Rule,* Ubuntu.

ANALYSIS AND CONFIGURATION NETWORK INTRUSION PREVENTION SYSTEM (NIPS) ON LINUX UBUNTU 10.04 LTS

DISKHAMS MAULIDI MYDZA

10451025514

Final Exam Date : *July 05th, 2011*

Graduation Ceremony Period : October 2011

Informatics Engineering Department
Faculty of Sciences and Technology
State Islamic University of Sultan Syarif Kasim Riau

ABSTRACT

Snort is one of open source Intrusion Detection System/IDS and used by network administrator as a network monitoring system as well as a detection for anomaly traffic on the network. Function of Snort as a Intrusion Detection System can be extended as a Intrusion Prevention System/IPS with help of firewall (IPTables) and Snortsam.

Snortsam is a plugin which make Snort rule able to order firewall (IPTables) for blocking data packet that detected as a intrusion by Snort. Snort identified data packet as a intrusion because data packet's pattern same with Snort rule's pattern that identify data packet as a intrusion. Log from intrusion detection saved as alert.

In this final task, author will configuration Snort as a Intrusion Prevention System/IPS combine with Snortsam and firewall (IPTables) on Ubuntu operating system. Choosing Ubuntu as a operating system on this Intrusion Prevention System configuration because Ubuntu is a Linux operating system which easy to used and easy to extended as a user desire.

Keywords *: firewall (IPTables), Intrusion Detection System, Intrusion Prevention System, Linux, Snort, Snortsam, Snort rule, Ubuntu.*

DAFTAR ISI

Halaman

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1 Latar Belakang

Teknologi komunikasi dan informasi telah berkembang dengan pesat dan memberikan pengaruh besar bagi kehidupan manusia, seperti perkembangan *internet* yang memungkinkan pengguna (*user*) untuk saling bertukar data. Keterbukaan akses tersebut memudahkan pengguna untuk saling berkomunikasi antar pengguna-pengguna yang lain di seluruh belahan dunia.

Dengan perkembangan teknologi tersebut, kejahatan teknologi komunikasi dan informasi juga ikut berkembang. Keamanan data pada komputer yang terhubung dalam sebuah jaringan sangat penting untuk dijaga validitas dan integritasnya serta dijamin ketersediaannya bagi pengguna. Sistem harus dilindung dari pengguna yang tidak berhak mengaksesnya (*unauthorized user*).

Sistem pertahanan komputer pada jaringan terhadap aktivitas gangguan saat ini umumnya dilakukan secara manual oleh *network administrator*. Hal ini mengakibatkan keamanan sistem bergantung pada kecekatan dan kecepatan *administrator* dalam merespon gangguan. Apabila gangguan tersebut dapat membuat suatu jaringan mengalami *malfunction*, seorang *network administrator* tidak dapat lagi mengakses sistem secara *remote* sehingga tidak akan dapat melakukan pemulihan sistem dengan cepat.

Untuk mengatasi masalah diatas, dibutuhkan sebuah sistem yang mampu mendeteksi penyusupan dalam jaringan komputer yang dikenal dengan *intrusion detection system* (IDS). IDS memiliki kemampuan untuk mendeteksi tindakan-tindakan yang mencurigakan di dalam sebuah jaringan komputer dan memiliki peran yang sangat besar dalam mengamankan sebuah jaringan komputer. Banyak *network administrator* menggunakan sistem IDS ini, salah satunya adalah Snort yang merupakan IDS yang bersifat *open source*. Namun Snort hanya mampu melakukan pendeteksian serangan melalui analisa dari paket data yang lewat pada jaringan dan jika terdapat keanehan pada suatu paket data maka akan dianggap sebuah penyerangan dan memberikan peringatan kepada *network administrator*.

Firewall juga menjadi solusi dalam mengamankan jaringan komputer saat ini tetapi kemampuan *firewall* hanya bisa menyaring paket data yang akan masuk ke dalam sistem.

Oleh karena itu dibutuhkan suatu sistem yang dapat menanggulangi ancaman keamanan yang mungkin terjadi secara optimal dalam waktu yang cepat dan secara otomatis langsung mengatasi penyusupan tersebut sehingga kemungkinan kerusakan akibat penyusupan keamanan jaringan dapat lebih diminimalisir. Sistem pencegahan penyusupan (*Intrusion Prevention System*) adalah sebuah sistem yang bekerja secara otomatis untuk memonitor kejadian pada jaringan komputer dan dapat mengatasi masalah keamanan jaringan. Snort sebagai sistem pendeteksi penyusupan bisa dikembangkan fungsinya menjadi sistem pencegahan penyusupan dengan bantuan *plugin* Snortsam dan *firewall*. *Firewall* yang digunakan adalah *firewall* bawaan dari sistem operasi Linux Ubuntu, yaitu IPTables. Penggunaan *firewall* IPTables sebagai aplikasi pendukung dalam sistem pencegahan penyusupan karena IPTables mudah dikonfigurasikan bersama *rule* Snort untuk mem-*block* paket data yang berupa serangan penyusupan. Selain itu, Linux Ubuntu merupakan sistem operasi *free* dan *open source* yang mudah digunakan, dikonfigurasikan dan dikembangkan sesuai dengan keinginan dari penggunanya.

Pada tugas akhir ini akan dikembangkan fungsi Snort sebagai sistem pencegahan penyusupan pada sistem operasi Linux Ubuntu 10.04 LTS.

1.2 Rumusan Masalah

Permasalahan yang dirumuskan berdasarkan latar belakang di atas adalah bagaimana menganalisa dan mengembangkan Snort sebagai sistem pendeteksian penyusupan (*Intrusion Detection System*) menjadi sebuah sistem yang mampu melakukan pencegahan penyusupan (*Intrusion Prevention System*) dengan cara mengkonfigurasikan Snort, Snortsam dan *firewall* (IPTables) pada sistem operasi Linux Ubuntu 10.04.

1.3 Batasan Masalah

Batasan masalah dari penyusunan tugas akhir ini adalah:

1. Menggunakan Snort sebagai sistem pendeteksian penyusupan.
2. Mengkonfigurasikan Snort menjadi sistem yang mampu melakukan pencegahan penyusupan dengan bantuan *plugin* Snortsam dan *firewall* (IPTables).
3. Menggunakan Linux Ubuntu 10.04 sebagai sistem operasi untuk melakukan konfigurasi Snort.

1.4 Tujuan Penelitian

Penelitian yang dilakukan pada tugas akhir ini bertujuan untuk menganalisa dan mengkonfigurasikan Snort pada sistem operasi Linux Ubuntu sebagai sistem yang melakukan pencegahan penyusupan pada jaringan (*Network Intrusion Prevention System*) yang mendeteksi serangan penyusupan dan juga melakukan pencegahan terhadap serangan penyusupan tersebut.

1.5 Manfaat Penelitian

Manfaat penelitian yang dilakukan adalah:

1. Mampu mengkonfigurasikan Snort sebagai sistem pencegah penyusupan pada komputer.
2. Mampu menjaga akses jaringan dari pihak yang tidak berhak mengaksesnya (*unauthorized user*).

1.6 Sistematika Penulisan

Sistematika penulisan laporan tugas akhir terdiri dari enam bab. Penjelasan mengenai enam bab ini, yaitu:

BAB I PENDAHULUAN

Bab ini menjelaskan dasar-dasar dari penulisan laporan tugas akhir ini, yang terdiri dari latar belakang, rumusan masalah, batasan masalah,

tujuan penelitian, manfaat penelitian dan sistematika penulisan laporan tugas akhir.

BAB II LANDASAN TEORI

Bab ini membahas teori-teori yang berhubungan dengan spesifikasi pembahasan penelitian yang akan diangkat, yang terdiri dari pembahasan mengenai Pengenalan Jaringan Komputer, *Protocol*, *Transmission Control Protocol/Internet Protocol* (TCP/IP), *Model Open System Interconnection* (OSI) *Layer*, *Media Access Control* (MAC) *Address*, *Address Resolution Protocol* (ARP), Ancaman Keamanan Jaringan, *Intrusion Detection System* (IDS), *Intrusion Prevention System* (IPS), Sistem Operasi, Snort dan IPTables.

BAB III METODOLOGI PENELITIAN

Bab ini membahas Studi Pustaka, Pemilihan Komponen yang Digunakan oleh Sistem, Implementasi dan Pengujian.

BAB VI ANALISA DAN KONFIGURASI

Bab ini membahas Analisa Sistem yang Sedang Berjalan, Analisa Sistem yang Akan Dikembangkan dan Konfigurasi Sistem Pencegah Penyusupan Pada Jaringan (*Network Intrusion Prevention System*).

BAB V IMPLEMENTASI DAN PENGUJIAN

Bab ini berisi tentang Implementasi Sistem, Pengujian *Network Intrusion Prevention* Sistem dan Kesimpulan Pengujian.

BAB VI PENUTUP

Bab ini berisi kesimpulan yang dihasikan dari pembahasan tentang sistem yang dikembangkan dan beberapa saran yang berisi perbaikan atas apa yang menjadi kekurangan dalam implementasi sehingga menjadi acuan bagi pengembangan selanjutnya.

BAB II

LANDASAN TEORI

Landasan teori disusun berdasarkan teori-teori yang berhubungan dengan Pengenalan Jaringan Komputer, *Protocol*, *Transmission Control Protocol/Internet Protocol* (TCP/IP), *Model Open System Interconnection* (OSI) *Layer*, *Media Access Control* (MAC) *Address*, *Address Resolution Protocol* (ARP), Ancaman Keamanan Jaringan, *Intrusion Detection System* (IDS), *Intrusion Prevention System* (IPS), Sistem Operasi, Snort dan IPTables.

2.1 Pengenalan Jaringan Komputer

2.1.1 Definisi Jaringan

Jaringan komputer merupakan sekelompok komputer atau *pheriperal* lain yang saling berhubungan antara satu dengan lainnya menggunakan *protocol* komunikasi melalui media komunikasi sehingga dapat saling berbagi sumber daya, serta dapat dikendalikan oleh suatu komputer pusat (Rahmat Rafiudin, 2003).

2.1.2 Jenis Jaringan

Berdasarkan ruang lingkup jangkauannya, maka jenis-jenis jaringan komputer dapat dibedakan menjadi 4 jenis, yaitu:

1. *Local Area Network* (LAN)

   Merupakan jenis jaringan *local* pada suatu area tertentu dengan jangkauan dan area terbatas. Misalnya jaringan antar ruang pada suatu kantor.

2. *Metropolitan Area Network* (MAN)

   Merupakan pengembangan dari jaringan LAN yang memiliki kecepatan transfer data yang tinggi, yang menghubungkan berbagai lokasi seperti kampus, perkantoran, pemerintahan dan sebagainya.

3. *Wide Area Network* (WAN)

   WAN dirancang untuk menghubungkan komputer-komputer yang terletak pada suatu cakupan geografis yang luas, seperti dari satu kota ke kota lain di dalam suatu negara.

4. *Global Area Network* (GAN)

   Suatu jaringan yang menghubungkan negara-negara di seluruh dunia.

Berdasarkan cara pengaksesan data, jaringan komputer dibagi menjadi:

1. *Client-Server* : jenis jaringan yang membagi fungsi komputer menjadi dua, ada komputer yang bertindak sebagai *server* dan ada komputer yang berperan sebagai *client* (*workstation*). Komputer *server* dapat mengontrol sepenuhnya komputer *client*.
2. *Peer to peer* (P2P) : jenis jaringan yang tidak membutuhkan *server* secara khusus karena komputer yang terhubung pada jaringan dapat berfungsi ganda sebagai *server* maupun *client*.

2.1.3 Komponen Dasar Jaringan

Sebelum membangun sebuah jaringan, ada beberapa komponen yang harus diperhatikan (Adi Waskita, 2004):

1. Komponen Fisik
   a. Komputer/PC
   b. Kartu jaringan atau *Network Interface Card* (NIC)
   c. Kabel jaringan (*twisted pair*, *coaxial*, *fiber optic*)
   d. Konektor : RJ-45 (*twisted pair*), BNC/T (*coaxial*), ST (*fiber optic*).
   e. HUB/Switch
   f. Router
   g. Modem (*Modem Demodulation*)
2. Komponen non-Fisik
   a. Sistem Operasi (*Operating System*)
   b. Protokol Jaringan (TCP/IP)

2.1.4 Topologi Jaringan Komputer

Topologi jaringan merupakan arsitektur komputer jaringan (*network architecture*) atau konfigurasi tentang bagaimana suatu jaringan disusun sedemikian rupa sehingga komputer dapat saling terhubung dalam sebuah jaringan yang telah dibuat. Topologi jaringan yang umum digunakan:

1. *Topologi Bus* : komputer *server* dan *client* dihubungkan secara berantai melalui kabel tunggal.

2. *Topologi Ring* : mirip dengan topologi bus namun bedanya topologi ring saling berhubungan dengan membentuk lingkaran yang seolah-olah berbentuk cincin.
3. *Topologi Star* : setiap komputer dihubungkan secara langsung melalui media perantara berupa hub atau switch.

2.2 *Protocol*

Protocol adalah sebuah aturan atau standar yang mengatur atau mengizinkan terjadinya hubungan, komunikasi, dan perpindahan data antara dua atau lebih titik komputer. *Protocol* dapat diterapkan pada perangkat keras, perangkat lunak atau kombinasi dari keduanya. Pada tingkatan terendah, *protocol* mendefinisikan koneksi perangkat keras. Prinsip dalam membuat *protocol* ada tiga hal yang harus dipertimbangkan yaitu efektivitas, kehandalan dan kemampuan dalam kondisi gagal di jaringan (Adi Waskita, 2004).

2.3 *Transmission Control Protocol/Internet protocol* (TCP/IP)

Transmission Control Protocol/Internet Protocol (TCP/IP) dikembangkan sebelum model OSI ada. Namun demikian lapisan-lapisan TCP/IP tidaklah cocok seluruhnya dengan lapisan-lapisan OSI. *Protocol* TCP/IP hanya dibuat atas lima lapisan saja, yaitu : *physical, data link, network, transport* dan *application*. Hanya lapisan aplikasi pada TCP/IP mencakupi tiga lapisan OSI teratas. Khusus layer ke empat, *protocol* TCP/IP mendefinisikan 2 buah *protocol* yakni *Transmission Control Protocol* (TCP) dan *user datagram protocol* (UDP). Sementara itu pada lapisan ketiga, TCP/IP mendefinisikan sebagai *internet Protocol* (IP), namun ada beberapa *protocol* lain yang mendukung pergerakan data pada lapisan ini (Adi Waskita, 2004).

Susunan lapisan pada TCP/IP yaitu:

1. *Physical Layer*

   Pada lapisan ini TCP/IP tidak mendefinisikan *protocol* yang spesifik. Artinya TCP/IP mendukung semua standar dan *proprietary protocol* lain. Pada lapisan ini ditentukan karakterisitik media transmisi, rata-rata

pensinyalan, serta skema pengkodean sinyal dan sarana sistem pengiriman data ke *device* yang terhubung ke jaringan.

2. *Data Link Layer*

   Berkaitan dengan *logical-interface* diantara satu ujung sistem dan jaringan dan melakukan fragmentasi atau defragmentasi *datagram*.

3. *Network layer*

   Berkaitan dengan *routing* data dari sumber ke tujuan. Pada lapisan ini TCP/IP mendukung IP dan didukung oleh *protocol* lain yaitu IP, ARP, RARP, ICMP, dan IGMP.

   1. *Internetworking Protocol* (IP) adalah mekanisme transmisi yang digunakan oleh TCP/IP. IP disebut juga *unreliable* dan *connectionless datagram protocol a besteffort delivery service*. IP mentransportasika data dalam paket-paket yang disebut *datagram*.
   2. *Address Resolution Protocol* (ARP) digunakan untuk menyesuaikan alamat IP dengan alamat fisik (*Physical Address*).
   3. *Reverse Address Resolution Protocol* (RARP) memperbolehkan *host* menemukan alamat IP nya jika dia sudah tahu alamat fisiknya. Ini berlaku pada *host* baru terkoneksi ke jaringan.
   4. *Internet Control Message Protocol* (ICMP) adalah suatu mekanisme yang digunakan oleh sejumlah *host* dan *gateway* untuk mengirim notifikasi *datagram* yang mengalami masalah kepada *host* pengirim.
   5. *Internet Group Message Protocol* (IGMP) digunakan untuk memfasilitasi transmisi pesan yang simulta kepada kelompok/grup penerima.

4. *Transport Layer*

   Pada lapisan ini terbagi dua yaitu UDP dan TCP

   1. *User datagram Protocol* (UDP) adalah *protocol process-to-process* yang menambahkan hanya alamat *port, check-sum error control*, dan panjang informasi data dari lapisan di atasnya (*connectionless*).
   2. *Transmission Control Protocol* (TCP) menyediaka layanan penuh lapisan pengiriman untuk aplikasi. TCP juga dikatakan *protocol*

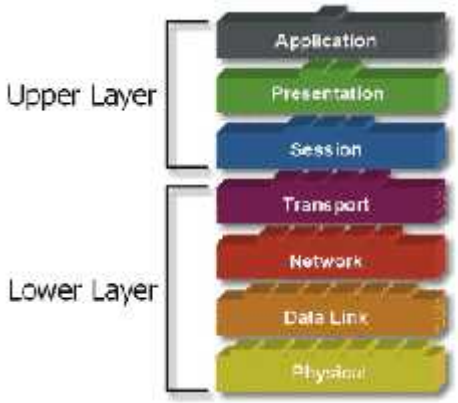
transport untuk *stream* yang *reliable*, dalam artian koneksi *end-to-end* harus dibangun terlebih dahulu di kedua ujung terminal sebelum kedua ujung terminal mengirim data (*connection oriented*).

5. *Application Layer*

   Layer dalam TCP/IP yang merupakan kombinasi lapisan-lapisan *session, presentation* dan *application* pada OSI yang menyediakan komunikasi diantara proses atau aplikasi-aplikasi pada *host* yang berbeda seperti : *telnet, ftp, http*, dan lain-lain.

2.4 *Model Open System Interconnection* (OSI) *Layer*

Model Open System Interconnection (OSI) diciptakan oleh *International Organization for Standardization* (ISO) yang menyediakan kerangka logika terstruktur bagaimana proses komunikasi data berinteraksi melalui jaringan. Standar ini dikembangkan untuk industri komputer agar komputer dapat berkomunikasi pada jaringan yang berbeda secara efisien (Adi Waskita, 2004).



Gambar 2.1 Model Layer OSI

Model dibagi menjadi 7 *layer*, dengan karakteristik dan fungsi masing-masing. Tiap *layer* harus dapat berkomunikasi dengan layer di atasnya maupun di bawahnya secara langsung melalui serentetan *protocol* dan *standard*.

Tabel 2.1 Keterangan OSI *Layer*

OSI Layer	Keterangan
Application	*Application Layer* : menyediakan jasa untuk aplikasi pengguna. *Layer* ini bertanggung jawab atas pertukaran informasi antara aplikasi-aplikasi komputer dan *service* lain yang berjalan di jaringan.
Application *Presentation*	*Presentation Layer* : bertanggung jawab bagaimana data dikonversi dan diformat untuk transfer data. Contoh konversi format *text* ASCII untuk dokumen, gif dan jpg untuk gambar. *Layer* ini membentuk kode konversi, transisi data, dan enkripsi.
Application *Presentation* *Session*	*Session Layer* : menentukan bagaimana dua terminal menjaga, memelihara dan mengatur koneksi, bagaimana mereka saling berhubungan satu sama lain.
Application *Presentation* *Session* *Transport*	*Transport Layer* : bertanggung jawab membagi data menjadi segmen, menjaga koneksi logika *end-to-end* antar terminal dan menyediakan penanganan error (*error handling*).
Application *Presentation* *Session* *Transport* *Network*	*Network Layer* : bertanggung jawab menentukan alamat jaringan, menentukan rute yang harus diambil selama perjalanan dan menjaga antria trafik di jaringan. Data pada *layer* ini berbentuk paket.
Application *Presentation* *Session* *Transport* *Network* *Data Link*	*Data Link Layer* : menyediakan *link* untuk data, memaketkannya menjadi *frame* yang berhubungan dengan *hardware* kemudian diangkut melalui *media*. Komunikasinya menggunakan kartu jaringan, mengatur komunikasi *layer physical* antara sistem koneksi dan penanganan eror.

Tabel 2.1 Keterangan OSI *Layer* (Lanjutan)

OSI *Layer*	Keterangan
Application *Presentation* *Session* *Transport* *Network* *Data Link* *Physical*	*Physical Layer* : bertanggung jawab atas proses data menjadi bit dan mentransfernya melalui *media*, seperti kabel dan menjaga koneksi fisik antar sistem.

2.5 *Media Access Control* (MAC) *Address*

Media Access Control (MAC) *Address* adalah sebuah alamat jaringan yang diimplementasikan pada lapisan *data link layer* pada OSI *layer* yang merepresentasikan sebuah node tertentu dalam jaringan. Dalam sebuah jaringan *ethernet*, MAC *address* merupakan alamat yang unik yang memiliki panjang 48 *bit* (6 *byte*) yang mengidentifikasikan sebuah komputer, interface dalam sebuah *router*, atau node lainnya dalam jaringan. MAC *address* juga sering disebut *ethernet address, physical address* atau *hardware address*.

MAC *address* mengizinkan perangkat-perangkat dalam jaringan agar dapat berkomunikasi antara satu dengan yang lainnya. Dalam sebuah komputer, MAC *address* ditetapkan ke sebuah kartu jaringan (*network interface card*) yang digunakan untuk menghubungkan komputer yang bersangkutan ke jaringan. MAC *address* umumnya tidak dapat diubah karena telah dimasukkan ke dalam *Read Only Memory* (ROM).

MAC *address* memang harus unik dan untuk itulah *Institute of Electrical and Electronics Engineers* (IEEE) mengalokasikan blok-blok dalam MAC *address*. 24 *bit* pertama dari MAC *address* merepresentasikan pembuat kartu jaringan dan 24 *bit* sisanya merepresentasikan nomor kartu tersebut. Berikut adalah tabel beberapa pembuat kartu jaringan dengan nomor identifikasi dalam MAC *address*.

Tabel 2.2 MAC *address* perusahaan pembuat kartu jaringan

Nama *Vendor*	**MAC *address***
Cisco Systems	00 00 0C
Cabletron Systems	00 00 1D
International Business Machine Corporation	00 04 AC
3*Com Corporation*	00 20 AF
GVC Corporation	00 C0 A8
Apple Computer	08 00 07
Hawlett-Packard Company	08 00 09

2.6 *Address Resolution Protocol* (ARP)

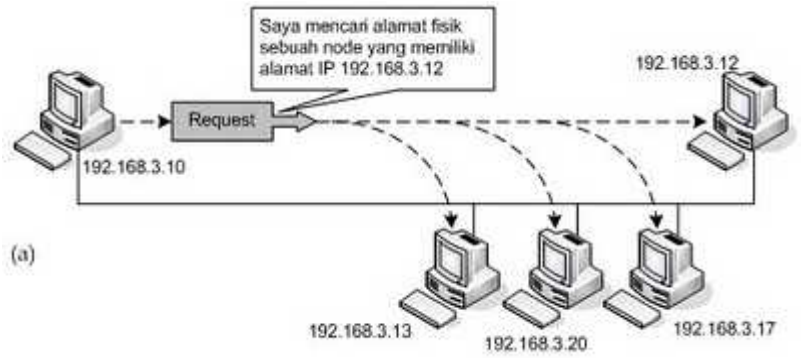
Address Resolution Protocol (ARP) adalah sebuah *protocol* dalam TCP/IP yang bertanggung jawab dalam melakukan resolusi IP *address* ke dalam MAC *address.*

Ketika sebuah aplikasi yang mendukung teknologi TCP/IP mencoba untuk mengakses sebuah *host* TCP/IP dengan menggunakan IP *address*, maka IP *address* yang dimiliki oleh *host* yang dituju harus diterjemahkan terlebih dahulu ke dalam MAC *address* agar *frame-frame* data dapat diteruskan ke tujuan dan diletakkan di atas *media* transmisi, setelah diproses terlebih dahulu oleh kartu jaringan (*network interface card*). Hal ini dikarenakan kartu jaringan beroperasi dalam *physical layer* dan *data link layer* pada OSI *layer* dan menggunakan *physical address* dari pada *logical address* (seperti IP *address*) untuk melakukan komunikasi data dalam jaringan.

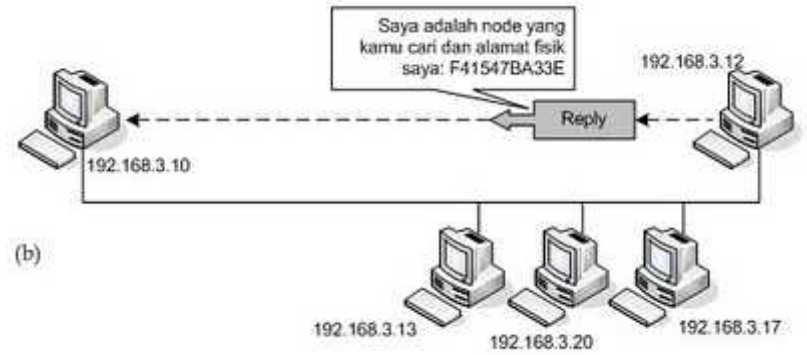
Dalam komunikasinya, ARP memanfaatkan 4 pesan (*message*), yaitu:

1. ARP *Request* : pesan ini digunakan untuk meminta MAC *address* dari suatu IP *address*. Pesan ini biasanya di *broadcast* ke semua *host* pada jaringan melalui *broadcast ethernet address*.
2. ARP *Reply* : jawaban dari ARP *request*. Setiap *host* yang menerima ARP *request* akan memeriksa permintaan tesebut untuk mengetahui apakah dirinya adalah pemilik IP *address* yang ada di dalamnya. ARP *reply* berisi MAC *address* dari IP *address* yang diminta.

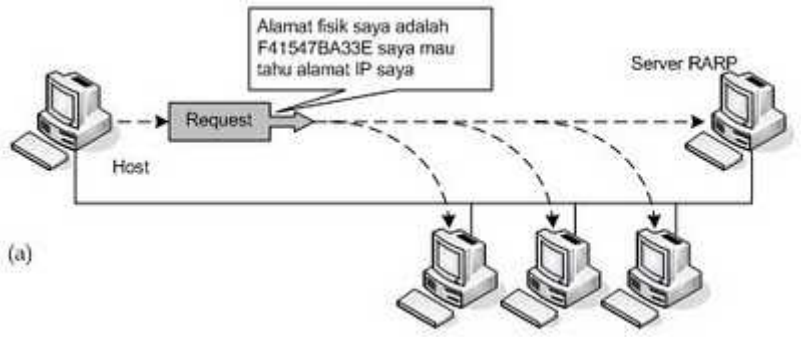
3. RARP *Request* : pesan ini meminta IP *address* dari suatu MAC *address.*
4. RARP *Reply* : pesan ini merupakan jawaban dari RARP *request*. Berisi IP *address* dari MAC *address* yang diminta.



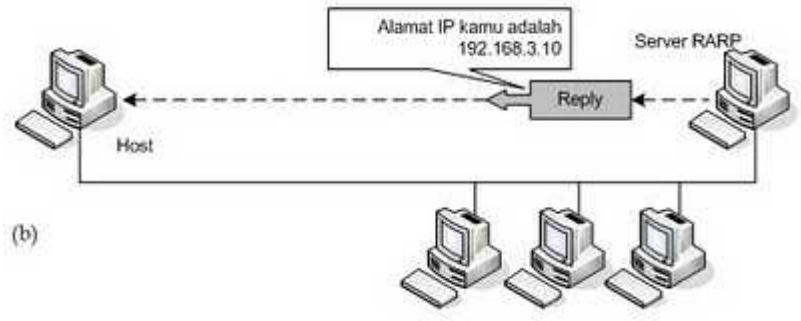
Gambar 2.2 analogi ARP *request*



Gambar 2.3 analogi ARP *reply*



Gambar 2.4 analogi RARP *request*



Gambar 2.5 analogi RARP *reply*

2.7 Ancaman Keamanan Jaringan

Salah satu pusat perhatian dalam keamanan jaringan adalah mengendalikan akses terhadap *resources* jaringan.

2.7.1 Prinsip Keamanan Jaringan

Prinsip keamanan jaringan sering disebut dengan segitiga CIA (*Confidentiality, Integrity, Availability*).

1. *Confidentiality* (Kerahasiaan) : menjaga infrastruktur jaringan agar tidak dapat diakses oleh pihak yang tidak berhak untuk mengaksesnya.
2. *Integrity* (Integritas) : menjaga data agar tetap asli, tidak dimodifikasi dalam perjalanannya dari sumber menuju penerimanya.
3. *Availability* (Ketersediaan) : *user* yang mempunyai hak akses (*authorized user*) diberi akses tepat waktu dan tidak terkendala apapun.

2.7.2 Jenis-Jenis Serangan Terhadap Keamanan Jaringan

Berikut ini adalah berbagai macam kelas serangan atau metode serangan terhadap keamanan infrastruktur jaringan.

2.7.2.1 *Brute Force and Dictionary*

Serangan ini adalah upaya untuk masuk ke dalam jaringan dengan menyerang *database password* atau menyerang *login prompt* yang sedang aktif. *Brute force* menggunakan cara yang sistematis dengan mencoba berbagai kombinasi angka, huruf, simbol dan kamus data yang telah didefenisikan terlebih dahulu untuk menemukan *password* dari *account user*.

2.7.2.2 *Denial of Service* (DoS)

DoS adalah salah satu ancaman keamanan jaringan yang membuat suatu layanan jaringan jadi tersendat, serangan yang membuat jaringan tidak bisa diakses atau serangan yang membuat sistem tidak bisa memproses atau merespon permintaan layanan terhadap *object* dan *resource* jaringan. Bentuk umum dari serangan DoS ini adalah dengan cara mengirim paket data dalam jumlah yang sangat bersar terhadap suatu *server* dimana *server* tersebut tidak bisa memproses semuanya. Bentuk lain dari serangan DoS ini adalah memanfaatkan *port-port* yang rentan dari sistem operasi. Tidak semua DoS merupakan akibat dari serangan keamanan jaringan. Kesalahan dalam *coding* suatu program juga bisa mengakibatkan kondisi seperti serangan DoS. Ada beberapa jenis dari DoS, antara lain:

1. *Distributed Denial of Service* (DDoS)

   Terjadi saat penyerang berhasil menggabungkan beberapa layanan sistem dan menggunakannya sebagai pusat untuk menyebarkan serangan terhadap korban.

2. *Distributed Reflective Denial of Service* (DRDoS)

   Memanfaatkan operasi normal dari layanan internet seperti *protocol-protocol update* DNS dan router. DRDoS ini menyerang fungsi dengan mengirim *update* dalam jumlah yang sangat besar kepada berbagai macam layanan *server* atau router dengan menggunakan *address spoofing* kepada target korban.

3. SYN *flooding*

   Upaya untuk membanjiri sinyal SYN kepada sistem yang menggunakan *protocol* TCP/IP dalam melakukan inisiasi sesi komunikasi.

4. *Smurf Attack*

   Server digunakan untuk membanjiri korban dengan data sampah yang tidak berguna. *Server* atau jaringan yang dipakai menghasilkan respon paket yang banyak seperti ICMP ECHO paket atau UDP paket dari satu paket yang dikirim.

5. *Ping of Death*

   Dengan menggunakan *tool* khusus, penyerang dapat mengirimkan paket *ping* yang *oversize* yang banyak kepada korban. *Ping of death* tidak lebih dari semacam serangan *buffer overflow*. Serangan ini dapat menyebabkan *crash* sistem, *freeze* atau *reboot*.

6. *Stream Attack*

   Serangan ini terjadi saat banyak jumlah paket yang besar dikirim menuju ke *port* pada sistem korban menggunakan sumber nomor yang random.

2.7.2.3 ARP *Spoofing*

Spoofing adalah seni untuk menjelma menjadi sesuatu yang lain. *Spoofing attack* terdiri dari IP *address* dan *node source* yang asli diganti dengan IP *address* dan *node source* yang lain.

ARP *Spoofing* muncul ketika ada upaya manipulasi terhadap pengalamatan nomor IP dan MAC *address*. Dengan ARP *spoofing*, komputer yang digunakan untuk penyerangan dapat memanipulasi lalu lintas data dari komputer-komputer *client* yang saling berhubungan dengan komputer *server* agar semua paket-paket *data* komputer *client* dapat melalui komputer penyerang terlebih dahulu. Proses ini terjadi karena komputer yang digunakan untuk serangan akan memberi tahu kepada komputer *client* yang menjadi target serangan bahwa MAC *address* komputer penyerang adalah MAC *address* dari komputer *server* dan memberi tahu komputer *server* bahwa MAC *adress* dari komputer penyerang adalah MAC *address* dari komputer *client* yang akan dituju oleh komputer *server*. Jenis serangan ini sering disebut dengan *Men-in-the-Middle* (MiTM). Dengan tipe serangan ini, penyerang dapat menyadap semua paket data yang dikirim oleh komputer *client* ke komputer *server*. Serangan ini dapat digunakan untuk mencuri *account* (*username* dan *password*) dari komputer *client*.

Serangan ini dapat diantisipasi dengan cara mengecek secara manual dengan mengetik ‘arp’ tanpa tanda kutip di *command prompt* pada sistem operasi Windows atau di *console* pada sistem operasi Linux. Cara lain untuk mengantisipasi serangan ARP ini adalah dengan cara mengeset ARP *static* pada komputer, sehingga IP *address* yang tidak terdaftar tidak akan bisa melakukan

serangan ARP lagi. Tapi dua cara ini sulit dilakukan karena begitu banyaknya IP *address* yang saling berhubungan dengan komputer jika telah berada pada lingkung jaringan *internet* pada setiap harinya sehingga cara manual ini tidak efektif dalam mencegah serangan ARP.

2.7.2.4 *Man-in-The-Middle* (MiTM) *attacking*

Serangan ini terjadi saat *attacker* bertindak sebagai perantara diantara dua *node* yang saling berkomunikasi. *Attacker* tidak akan tampak pada kedua sisi *node* tersebut dan dapat melihat atau mengubah isi dari *traffic*.

2.7.2.5 *Sniffer*

Merupakan kegiatan untuk mendapatkan informasi tentang *traffic* jaringan. Suatu *sniffer* sering merupakan program penagkap paket yang bisa menduplikasikan isi paket yang lewat pada media jaringan ke dalam *file*. Serangan ini sering difokuskan pada koneksi awal antara *client* dan *server* untuk mendapatkan *account user* dan lainnya.

2.7.2.6 *Spamming*

Spam pada umumnya bukan merupakan serangan keamanan jaringan akan tetapi hamper mirip dengan DoS. *Spam* bisa berupa iklan atau trojan.

2.7.2.7 *Scanning*

Scanning terbagi atas tiga jenis, yaitu:

1. *Port Scanning* : merupakan kegiatan *scanning* yang bertujuan menemukan *port-port* yang terbuka dari suatu *host.*
2. *Network Scanning* : merupakan kegiatan *scanning* yang bertujuan menemukan *host* atau komputer yang aktif pada suatu jaringan.
3. *Vulnerability Scanning* : merupakan kegiatan *scanning* yang bertujuan menemukan kelemahan dari sebuah sistem.

2.8 *Intrusion Detection System* (IDS)

Intrusion Detection System (IDS) merupakan suatu perangkat lunak atau sistem perangkat keras yang bekerja secara otomatis untuk memonitor kejadian pada jaringan computer dan dapat menganalisa masalah keamanan jaringan.

Suatu IDS dapat didefenisikan sebagai *tool*, metode atau sumber daya yang memberikan bantuan untuk melakukan identifikasi, memberikan laporan terhadap aktivitas jaringan komputer. IDS sebenarnya tidak mendeteksi penyusup tetapi hanya mendeteksi aktivitas *traffic* jaringan yang tidak layak terjadi sehingga awal dari langkah kerja penyerang bisa diketahui. Dengan demikian *network administrator* dapat melakukan tindakan pencegahan dan bersiap atas kemungkinan serangan yang akan terjadi (Raven Alder, 2004).

Ada 2 jenis IDS, yaitu *Host Based Intrusion Detection System* (HIDS) dan *Network Based Intrusion Detection System* (NIDS).

2.8.1 *Host Intrusion Detection System* (HIDS)

HIDS bekerja pada *host* yang akan dilindungi. IDS jenis ini dappat melakukan berbagai macam tugas untuk mendeteksi serangan yang dilakukan pada *host* tersebut. Keunggulan HIDS adalah tugas-tugas yang berhubungan dengan keamanan *file*. Misalnya ada tidaknya *file* yang telah diubah atau ada usaha untuk mendapatkan akses ke *file-file* yang sensitif.

2.8.2 *Network Intrusion Detection System* (NIDS)

Digunakan untuk melakukan monitoring di seluruh segmen jaringan. NIDS akan mengumpulkan paket-paket data yang terdapat pada jaringan kemudian menganalisanya serta menentukan apakah paket-paket tersebut berupa suatu paket yang normal atau suatu aktivitas yang mencurigakan.

2.9 *Intrusion Prevention System* (IPS)

Intrusion Prevention System (IPS) adalah sebuah perangkat lunak atau perangkat keras yang bekerja untuk *monitoring* trafik jaringan, mendeteksi aktivitas yang mencurigakan dan melakukan pencegahan dini terhadap penyusupan atau kejadian yang dapat membuat jaringan menjadi berjalan tidak seperti sebagaimana mestinya. IPS merupakan pendekatan yang sering digunakan untuk membangun sistem keamanan komputer, IPS mengombinasikan teknik *firewall* dan metode *intrusion detection system* (IDS) dengan sangat baik. Teknologi ini dapat digunakan untuk mencegah serangan yang akan masuk ke jaringan lokal dengan memeriksa dan mencatat semua paket data serta mengenali

paket dengan sensor saat seragan teridentifikasi. Jadi IPS bertidak seperti layaknya *firewall* yang akan mengizinkan atau menghalang paket data (Raven Alder, 2007).

Secara khusus, IPS memiliki empat komponen utama, yaitu:

1. *Normalisasi Traffic* : menginterpretasikan *traffic* jaringan dan melakukan analisa terhadap paket yang disusun kembali, seperti halnya fungsi *block* sederhana.
2. *Detection Engine* : mendeteksi *traffic* jaringan dan melakukan *pattern matching* terhadap tabel acuan dan respon yang sesuai.
3. *Service Scanner* : membangun suatu tabel acuan untuk mengelompokkan informasi.
4. *Traffic Shaper* : membentuk dan mengatur *traffic* jaringan.

Ada 2 jenis IPS, yaitu *Host Based Intrusion Prevention System* (HIPS) dan *Network Based Intrusion Prevention System* (NIPS).

2.9.1 *Host Intrusion Prevention System* (HIPS)

Host-based Intrusion Prevention System (HIPS) sama seperti halnya *Host Based Intrusion Detection System* (HIDS). Program agent HIPS diinstall secara langsung di sistem yang diproteksi untuk dimonitor aktifitas sistem internalnya. HIPS di binding dengan kernel sistem operasi dan *services* sistem operasi sehingga HIPS bisa memantau dan menghadang *system call* yang dicurigai dalam rangka mencegah terjadinya intrusi terhadap *host*. HIPS juga bisa memantau aliran data dan aktivitas pada aplikasi tertentu. Sebagai contoh HIPS untuk mencegah *intrusion* pada *webserver* misalnya. Dari sisi *security* mungkin solusi HIPS bisa mencegah datangnya ancaman terhadap *host*. Tetapi dari sisi *performance*, harus diperhatikan apakah HIPS memberikan dampak negatif terhadap *performance host*. Karena menginstall dan binding HIPS pada sistem operasi mengakibatkan penggunaan *resource* komputer *host* menjadi semakin besar.

2.9.2 *Network Intrusion Prevention System* (NIPS)

Network-based Intrusion Prevention System (NIPS) tidak melakukan pantauan secara khusus di satu host saja. Tetapi melakukan pantauan dan proteksi

dalam satu jaringan secara *global*. NIPS menggabungkan fitur IPS dengan *firewall* dan kadang disebut sebagai *In-Line* IDS atau *Gateway Intrusion Detection System* (GIDS).

Sistem kerja IPS yang populer yaitu pendeteksian berbasis *signature*, pendeteksian berbasis anomali, dan monitoring *file* pada sistem operasi *host*.

1. Sistematika IPS yang berbasis *signature* adalah dengan cara mencocokkan lalu lintas jaringan dengan *signature database* milik IPS yang berisi *attacking rule* atau cara-cara serangan dan penyusupan yang sering dilakukan oleh penyerang. Sama halnya dengan antivirus, IPS berbasis *signature* membutuhkan *update* terhadap *signature database* untuk metode-metode penyerangan terbaru. IPS berbasis *signature* juga melakukan pencegahan terhadap ancaman intrusi sesuai dengan *signature database* yang bersangkutan.
2. Sistematika IPS yang berbasis anomali adalah dengan cara melibatkan pola-pola lalu lintas jaringan yang pernah terjadi. Umumnya, dilakukan dengan menggunakan teknik statistik. Statistik tersebut mencakup perbandingan antara lalu lintas jaringan yang sedang di *monitor* dengan lalu lintas jaringan yang biasa terjadi (*normal state*). Metode ini dapat dikatakan lebih kaya dibandingkan *signature-based* IPS. Karena *anomaly-based* IPS dapat mendeteksi gangguan terhadap jaringan yang terbaru yang belum terdapat di *database* IPS. Tetapi kelemahannya adalah potensi timbulnya *false positive*, yaitu pesan/log yang belum semestinya dilaporkan. Sehingga tugas *Network Administrator* menjadi lebih rumit, dengan harus memilah-milah mana yang merupakan serangan yang sebenarnya dari banyaknya laporan *false positive* yang muncul.
3. Teknik lain yang digunakan adalah dengan cara melakukan monitoring berkas-berkas sistem operasi pada *host*. IPS akan melihat apakah ada percobaan untuk mengubah beberapa berkas sistem operasi, utamanya berkas log. Teknik ini diimplementasikan dalam IPS jenis *Host Based Intrusion Prevention System* (HIPS).

Teknik yang digunakan IPS untuk mencegah serangan ada dua, yaitu *sniping* dan *shunning*.

1. *Sniping* : memungkinkan IPS untuk menterminasi serangan yang dicurigai melalui penggunaan paket TCP RST atau pesan ICMP *Unreachable*.
2. *Shunning* : memungkinkan IPS mengkonfigurasi secara otomatis *firewall* untuk men-*drop traffic* berdasarkan apa yang dideteksi oleh IPS. Untuk kemudian melakukan *prevention* atau pencegahan terhadap koneksi tertentu.

2.10 Sistem Operasi

Sistem operasi adalah sebuah layer atau *software* yang bertugas untuk mengontrol *device* dan memberikan *user program* dengan *interface* sederhana terhadap *hardware.* Defenisi lain dari sistem operasi adalah sistem yang mampu melakukan pengontrolan yang bersifat *hardware oriented* yaitu pendekatan secara perangkat keras (Von Hagen, 2007).

2.10.1 Sistem Operasi Linux

Sistem operasi GNU Linux terdiri dari tiga bagian kode penting, yaitu:

1. *Kernell Linux*

   Kernell merupakan inti dari suatu sistem operasi yang bertugas sebagai pengendali dan mengontrol kinerja dari semua yang ada pada sistem, mulai dari peralatan, penggunaan memori untuk aplikasi yang sedang berjalan, mengatur peletakan *file*, mengenali *driver* dan hal lainnya.
2. *Library System*

   Library System menyediakan banyak tipe fungsi. Pada level paling rendah, mengizinkan aplikasi untuk melakukan permintaan pada *service* sistem *kernell.*
3. *Utility System*

   Sistem linux mengandung banyak program-program *user-mode*, *utilitity sistem* dan *utility user. Utilitity sistem* termasuk semua program yang diperlukan untuk menginisialisasi sistem, seperti program untuk konfigurasi alat jaringan (*network device*) atau untuk *load* modul kernel.

2.10.2 Ubuntu

Ubuntu merupakan distributor linux (*distro*) turunan dari debian. Tujuan dari *distro* Ubuntu adalah membawa semangat yang terkandung di dalam Ubuntu ke dalam dunia perangkat lunak. Ubuntu adalah sistem operasi lengkap berbasis Linux, tersedia secara bebas dan mempunyai dukungan baik yang berasal dari komunitas maupun tenaga ahli profesional. Ubuntu berkembang sangat cepat karena dukungan yang baik dan menyediakan berbagai kebutuhan aplikasi hampir di seluruh bidang komputer (Von Hagen, 2007).

Ubuntu memiliki beberapa kelebihan sebagai sistem operasi, diantaranya adalah:

1. Bersifat *open source*.
2. *User friendly*.
3. *Repository* lengkap

Sedangkan menurut beberapa pengguna ubuntu, ada beberapa kekurangan dari sistem operasi ini, diantaranya adalah:

1. Komunitas terbanyak hanya berada pada negara maju.
2. Tidak semua *driver* komputer didukung langsung oleh ubuntu.

2.11 Snort

Snort merupakan perangkat lunak untuk mendeteksi penyusup dan mampu menganalisa paket yang melintasi jaringan secara *real-time traffic* dan *logging* ke dalam *database* serta mampu mendeteksi berbagai serangan yang berasal dari luar jaringan (Jacob Babbin, 2005).

Snort bisa dioperasikan dengan tiga mode, yaitu:

1. Paket *Sniffer Mode*

   Snort bertindak sebagai *software sniffer* yang dapat melihat semua paket yang lewat dalam jaringan komputer dimana Snort dipasang. Dalam mode ini, berbagai paket ditampilkan hanya dalam bentuk layar monitor secara *real-time*.

   Beberapa contoh perintah pada mode ini, yaitu:

```
root@myubuntu:~# snort -v
```

```
root@myubuntu:~# snort -vd
```

```
root@myubuntu:~# snort -vde
```

Dengan menambahkan beberapa *switch* –v, -d, -e akan menghasilkan beberapa keluaran berbeda.

-v digunakan untuk melihat *header* TCP/IP paket yang lewat.

-d digunakan untuk melihat isi paket.

-e digunakan untuk melihat *header link layer* paket seperti *Ethernet header*.

2. Paket *Logger Mode*

   Dalam mode ini, selain dapat melihat semua paket yang lewat dalam jaringan komputer, Snort juga dapat mencatat atau melakukan *logging* terhadap berbagai paket tersebut ke dalam *database*. Dengan kata lain, Snort mampu membuat *copy* dari paket-paket yang lewat dan menyimpan *copy* tersebut ke dalam *database* sehingga *network administrator* dapat melakukan analisa terhadap paket data atau *traffic* jaringan tersebut.

   Perintah yang digunakan untuk mencatat paket yang lewat adalah:

   ```
   root@myubuntu:~# snort –dev –l ./log
   ```

   Perintah diatas akan melakukan pencatatan paket data ke dalam direktori log.

   ```
   root@myubuntu:~# snort –dv –r packet.log
   ```

   Perintah diatas digunakan untuk membaca data *logging* pada direktori log.

3. *Network Intrusion Detection System* (NIDS)

   Dalam mode ini, Snort dapat melakukan *monitoring* dan menganalisa paket data, mendeteksi adanya paket data yang didefinisikan sebagai sebuah serangan penyusupan dan melakukan *logging* terhadap serangan

penyusupan yang terjadi pada jaringan komputer berdasarkan *rule* yang telah ditetapkan oleh *network administrator.*

Perintah yang digunakan untuk mengaktifkan Snort yang digunakan dalam pendeteksian serangan penyusupan adalah:

```
root@myubuntu:~# snort –v -c /etc/snort/snort.conf –i eth0
```

Perintah diatas digunakan untuk melakukan *monitoring* paket data dan pendeteksian serangan penyusupan yang didefinisikan pada *file* konfigurasi bernama snort.conf

Layaknya sistem monitoring lainnya, Snort memiliki banyak kelebihan sekaligus kekurangan. Berikut adalah beberapa kelebihan SNORT sebagai *Intrusion Detection System*:

1. *Event Correlation*

   Memungkinkan *network administrator* melacak dan menghubungkan kejadian serangan penyusupan dengan cepat berdasarkan analisa *traffic* jaringan.
2. *Centralized Sensor Management*

   Kemampuan untuk menjalankan laporan yang detail pada keamanan jaringan.
3. *Signature Matching*

   Memonitor semua *traffic* yang melewati jaringan dan selanjutnya mencocokkan masing-masing paket dengan pola serangan yang telah didefenisikan pada *rule*. Selanjutnya Snort akan menghasilkan *alert* pemberitahuan serangan penyusupan kepada *network administrator.*
4. *Anomaly Detection*

   Mendeteksi kejanggalan pada *traffic* jaringan yang tidak biasa terjadi pada *traffic* jaringan normal.
5. *Customizable Signatures and Threshold*

   Memungkinkan *network administrator* membuat pola serangan pada *rule* Snort menangkal serangan yang mungkin terjadi.

6. *Lightweight*

   Snort tidak memakan banyak *resources* tetapi cukup canggih dan fleksibel untuk digunakan sebagai sistem *monitoring* pada jaringan dan mendeteksi adanya serangan penyusupan.

7. *Stealth Mode*

   Snort mampu menyembunyikan dirinya sendiri di dalam jaringan komputer sehingga keberadaannya tidak bisa terdeteksi oleh komputer manapun.

Meskipun Snort memiliki banyak fitur dengan berbagai kelebihan, tetap saja masih memiliki keterbatasan dalam fungsi yang disediakannya, diantaranya yaitu Snort belum mampu untuk mencegah serangan penyusupan jaringan secara langsung, dalam arti bahwa Snort adalah murni *tool* untuk me*monitoring* jaringan, mendeteksi serangan penyusupan dan memberikan *alert* pemberitahuan kepada *network administrator* ketika terjadi serangan tapi belum menyediakan fasilitas untuk pencegahan serangan penyusupan secara otomatis.

2.12 IPTables

IPTables merupakan sebuah fasilitas tambahan yang tersedia pada setiap perangkat komputer yang diinstal dengan sistem operasi Linux dan resmi diluncurkan pada *kernell* 2.4. Fitur ini harus diaktifkan terlebih dahulu saat melakukan kompilasi kernel untuk dapat menggunakannya. IPTables merupakan fasilitas tambahan yang memiliki tugas untuk menjaga keamanan perangkat komputer anda dalam jaringan. Atau dengan kata lain, IPTables merupakan sebuah firewall atau program IP filter *build-in* yang disediakan oleh *kernell* Linux untuk tetap menjaga agar perangkat tetap dalam keadaan aman (Raaven Alder, 2007).

Fitur yang dimiliki IPTables:

1. *Connection Tracking Capability* yaitu kemampuan untuk menginspeksi dan menyaring paket data pada TCP dan MAC *address* serta bekerja dengan ICMP dan UDP.

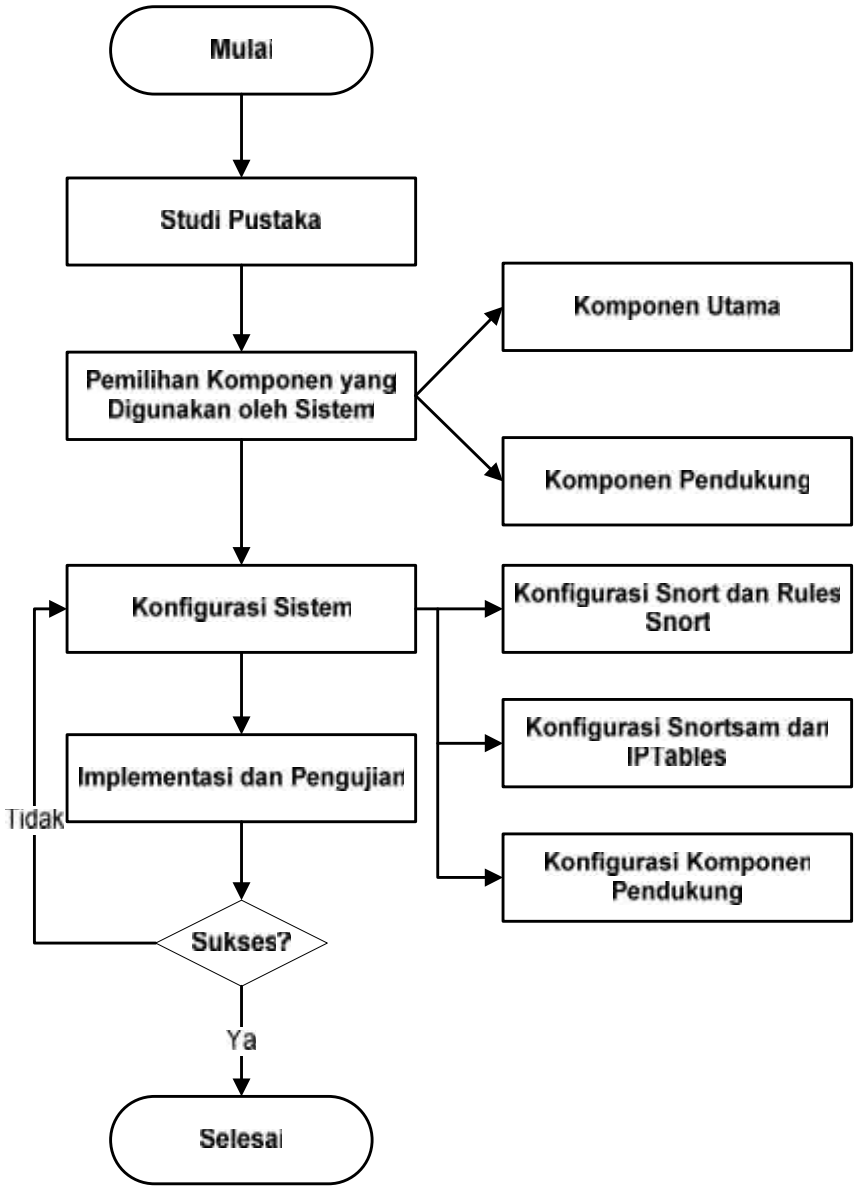
2. Menyederhanakan perilaku paket-paket dalam melakukan negosiasi *built in chain* (*input,output, dan forward*).

3. *Rate-Limited connection* dan *logging capability* yang bertujuan membatasi *traffic* sebagai tindakan pencegahan serangan *Syn flooding denial of services* (DOS).

BAB III

METODOLOGI PENELITIAN

Metodologi penelitian menguraikan seluruh kegiatan yang dilaksanakan selama kegiatan penelitian berlangsung dan digunakan sebagai pedoman dalam pelaksanaan penelitian agar hasil yang dicapai tidak menyimpang dari tujuan. Berikut tahapan penelitian :



Gambar 3.1 Tahapan Penelitian

3.1 Studi Pustaka

Studi Pustaka berguna untuk mendukung penelitian yang akan dikerjakan. Teori-teori bersumber dari *website*, buku, jurnal dan penelitian sejenis. Pada tahapan ini juga dilakukan analisa kebutuhan sistem yang akan dikonfigurasikan.

3.2 Pemilihan Komponen yang Digunakan oleh Sistem

Tahapan selanjutnya adalah melakukan pemilihan komponen utama dan komponen pendukung yang akan digunakan dalam pengkonfigurasian sistem. Tidak semua *versi* komponen yang dibutuhkan cocok untuk keperluan konfigurasi. Adapun komponen-komponen yang dibutuhkan adalah sebagai berikut:

1. Komponen utama
    a. Snort versi 2.9.0.5
    b. Snort *rule* versi 2905 *snapshot*
2. Komponen pendukung
    a. Snortsam
    b. ACIDBASE
    c. Apache2
    d. MySQL
    e. IPTables

Secara lengkap seluruh komponen dapat dilhat pada lampiran B.

3.3 Konfigurasi Sistem

Pada tahapan ini, komponen-komponen sistem yang dibutuhkan akan dikonfigurasikan menurut kesimpulan dan hasil analisa kebutuhan sistem yang telah dilakukan sebelumnya. Adapun tahap-tahap konfigurasinya adalah:

1. Tahap konfigurasi Snort dan komponen pendukung.
    a. Men-*download* Snort
    b. Men-*download* komponen pendukung
    c. Mengekstrak Snort dan komponen pendukung
    d. Menginstal dan mengkonfigurasikan Snort dan komponen pendukung

2. Tahap konfigurasi *rules* Snort.
    a. Men-*download rules* Snort
    b. Mengekstrak *rules* Snort
    c. Mengkonfigurasikan *rules* Snort dengan Snort yang digunakan
3. Tahap konfigurasi Snortsam
    a. Men-*download* Snortsam
    b. Mengesktrak dan mengkonfigurasikan Snortsam dengan Snort dan *rules* Snort
4. Tahap konfigurasi IPTables
    a. Mengaktifkan IPTables
    b. Mengkonfigurasikan IPTables dengan Snort

3.4 Implementasi dan Pengujian

Tahap implementasi adalah tahap dimana konfigurasi sistem dilakukan, melingkupi mulai dari kebutuhan perangkat keras (*hardware*) dan perangkat lunak (*software*) berdasarkan studi pustaka yang telah dilakukan sebelumnya dan melihat kecocokan komponen sistem yang dibutuhkan oleh sistem terhadap perangkat keras (*hardware*) dan perangkat lunak (*software*).

Lingkungan implementasi yang digunakan untuk mengkonfigurasikan Snort sebagai sistem pencegahan penyusupan pada jaringan (*Network Intrusion Prevention System*/NIPS) terdiri dari:

1. Perangkat Keras

    Perangkat keras yang digunakan memiliki spesifikasi sebagai berikut:

    a. *Processor* : Intel Core i5 2.4 GHz
    b. *Memory* : 2 GB
    c. *Harddisk* : 320 GB
2. Perangkat Lunak
    a. Sistem Operasi : Linux Ubuntu 10.04 LTS
    b. *Kernell* : 2.6.32-28-*generic*
    c. *Intrusion Detection System* : Snort 2.9.0.5
    d. *Rules* Snort : Snortrules-Snapshot-2905

e. Komponen Pendukung : Pada Lampiran B

f. *Output Plugin* : Snortsam

g. *Firewall* : IPTables

3. Lain-lain

a. Snort Sensor : eth0 – 192.168.1.1

b. *Range* IP *address* : 192.168.1.0/24

c. *Web Browser Monitoring* : ACIDBASE, Mozilla

Selanjutnya adalah pengujian (*testing*). Pada tahapan ini menggambarkan kondisi-kondisi yang terjadi apabila sistem dijalankan. Pengujian dilakukan dengan cara mengetes serangan penyusupan antara lain:

1. Pengujian *host scanning*
2. Pengujian *port scanning*
3. Pengujian *HTTP attacking*
4. Pengujian *SSH*
5. Pengujian *ping of death*

Lingkungan pengujian yang digunakan untuk menguji sistem pencegahan penyusupan pada jaringan (*Network Intrusion Prevention System*/NIPS) terdiri dari:

1. Perangkat Keras

a. *Processor* : Intel Core 2 Duo 1.73 Ghz

b. *Memory* : 2 GB

c. *Harddisk* : 120 GB

2. Perangkat Lunak

a. Sistem Operasi : Linux Backtrack 4 R2

b. Aplikasi Pengujian : Backtrack *tools*

c. IP *address* : 192.168.1.2

BAB IV

ANALISA DAN KONFIGURASI

Analisa adalah suatu proses identifikasi permasalahan dari kumpulan data yang bernilai informasi dan bermanfaat pada rancang bangun sistem. Analisa merupakan langkah awal dalam membuat suatu sistem. Analisa dilakukan untuk memahami persoalan sebelum melakukan tahap konfigurasi. Hal ini dilakukan untuk mencari kebutuhan-kebutuhan yang diperlukan oleh sistem dan kendala-kendala yang akan dicari solusinya. Setelah analisa dilakukan, tahap selanjutnya adalah konfigurasi Snort. Konfigurasi yang dibuat harus memiliki kesesuaian dengan analisa sistem yang sebelumnya telah dilakukan.

4.1 Analisa Sistem yang Sedang Berjalan

Intrusion Detection adalah sebuah metode yang dapat mendeteksi aktivitas yang mencurigakan pada level personal (*host*) dan jaringan (*network*). Salah satu perangkat lunak *open source* yang digunakan sebagai perangkat *Intrusion Detection* pada jaringan adalah Snort. Snort selain dapat me*monitoring* dan menganalisa paket data pada jaringan juga dapat mendeteksi aktifitas yang mencurigakan pada jaringan. Hasil deteksi tersebut disimpan dalam *log database*.

4.1.1 Komponen-komponen Snort *Intrusion Detection System* (IDS)

1. Snort *Engine*

   Snort *Engine* merupakan program yang selalu bekerja untuk membaca paket data dan kemudian membandingkan dengan *rule* Snort.

```
root@myubuntu:~# ps aux | grep snort
snort     2656  0.1  1.9  58376 38164 ?        Ss   23:08   0:00 /usr/sbin/snort -m
027 -D -d -l /var/log/snort -u snort -g snort -c /etc/snort/snort.conf -S
HOME_NET=[192.168.1.0/24] -i eth0
root      2847  0.0  0.0   3324   812 pts/0    S+   23:12   0:00 grep --color=auto
```

   Perintah diatas menunjukkan bahwa Snort *Engine* dalam keadaan aktif dengan proses ID 2847 dan dijalankan oleh *user* “root”.

2. *Rule* Snort

   Rule Snort merupakan *database* yang berisi pola-pola serangan berupa *signature* jenis-jenis serangan.

   Contoh *rule* Snort:

```
alert tcp $EXTERNAL_NET any <> $HOME_NET 0 (msg:"BAD-TRAFFIC
tcp port 0 traffic"; flow:stateless; classtype:misc-activity; sid:524; rev:8;)
alert udp $EXTERNAL_NET any <> $HOME_NET 0 (msg:"BAD-TRAFFIC
udp port 0 traffic"; reference:bugtraq,576; reference:cve,1999-0675;
reference:nessus,10074;
```

   Rule diatas terdiri dari 2 bagian yaitu : *header* dan *option.* Bagian "alert tcp $EXTERNAL_NET any <> $HOME_NET 0" dan "alert udp $EXTERNAL_NET any <> $HOME_NET 0" adalah *header* dan selebihnya merupakan bagian dari *option*. Dari *rule* Snort ini akan dikelompokkan apakah sebuah paket data yang lewat dianggap sebagai sebuah serangan penyusupan atau bukan.

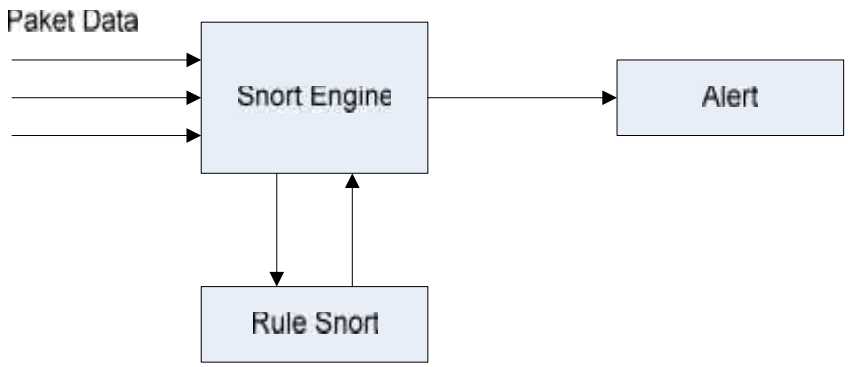
3. *Alert*

   Alert merupakan catatan serangan pada deteksi serangan penyusupan. Jika *rule* Snort mendefiniskan paket data yang lewat sebagai serangan penyusupan, maka Snort *Engine* akan mengirimkan *alert* berupa *log* ke dalam *database*.

```
06/13-12:45:56.208898  [**] [122:1:0] (portscan) TCP Portscan [**] [Priority:
3] {PROTO:255} 192.168.1.246 -> 192.168.1.244
06/13-12:45:56.208897  [**] [122:1:0] (portscan) TCP Portscan [**] [Priority:
3] {PROTO:255} 192.168.1.246 -> 192.168.1.244
```

   Contoh diatas merupakan *alert* hasil *scanning port* TCP dari IP 192.168.1.246 ke IP 192.168.1.244 yang disimpan oleh Snort *Engine* ke dalam *alert* dan dianggap sebagai sebuah serangan oleh Snort karena pola paket data tersebut terdapat dalam *rule* Snort.

Hubungan ketiga komponen Snort IDS diatas dapat digambarkan sebagai berikut:



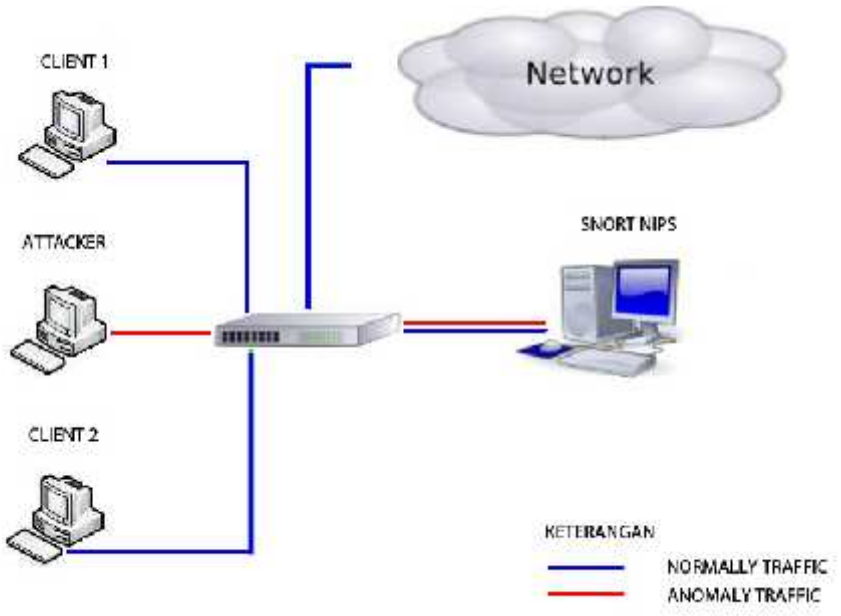
Gambar 4.1 Hubungan antara komponen-komponen Snort IDS

Meskipun Snort memiliki banyak fitur dengan berbagai kelebihan, tetap saja masih memiliki keterbatasan dalam fungsi yang disediakannya, diantaranya yaitu Snort belum mampu untuk mencegah serangan penyusupan jaringan secara langsung, dalam arti bahwa Snort adalah murni *tool* untuk me*monitoring* jaringan, mendeteksi serangan penyusupan dan memberikan *alert* pemberitahuan kepada *network administrator* ketika terjadi serangan tapi belum menyediakan fasilitas untuk pencegahan serangan penyusupan secara otomatis.

4.2 Analisa Sistem yang Akan Dikembangkan

Seperti yang telah dijelaskan sebelumnya, pada tugas akhir ini akan mengkonfigurasikan sebuah sistem yang memiliki kemampuan untuk me*monitoring* jaringan, mendeteksi adanya aktifitas yang mencurigakan di dalam jaringan yang didefinisikan sebagai serangan penyusupan dan melakukan pencegahan terhadap serangan penyusupan tersebut (*Intrusion Prevention System*). Sistem yang dikonfigurasikan merupakan pengembangan dari sistem yang sudah ada, yakni Snort

Skema dari sistem yang akan dibangun dapat digambarkan sebagai berikut:



Gambar 4.2 Skema *Network Intrusion Prevention System*

Rincian penjelasan skema sistem pencegah penyusupan pada jaringan dapat dilihat pada uraian dibawah ini:

1. *Attacker* berada pada satu jaringan bersama dengan *user/client* dan *server Intrusion Prevention System*.
2. *Attacker* melakukan serangan terhadap *server* maupun *user/client* yang berada pada jaringan.
3. *Server Intrusion Prevention System* merupakan suatu sistem yang memonitor *traffic* jaringan secara *real time*, tidak hanya pada *server Intrusion Prevention System* sendiri tapi juga memonitor aktifitas *traffic* jaringan pada *user/client*.
4. Jika terdapat aktifitas yang mencurigakan pada jaringan yang didefiniskan sebagai sebuah serangan penyusupan, *server Intrusion Prevention System* akan memberikan *alert* adanya gangguan pada jaringan dan secara

otomatis serangan penyusupan tersebut akan di-*block* oleh *firewall* (IPTables).

Untuk itu komponen-komponen yang harus ada pada sistem pencegahan penyusupan pada jaringan meliputi:

1. *Intrusion Detection System* (IDS)

   Dilihat dari cara kerja dalam menganalisa apakah paket dianggap sebagai penyusupan atau bukan, IDS dibagi menjadi 2 yaitu :

   a. *Knowledge-based* atau *misuse detection* yaitu mendeteksi adanya penyusupan dengan cara me-*monitoring* paket data kemudian membandingkannya dengan *rule* IDS yang berisi *signature* paket serangan. Jika paket data mempunyai pola yang sama dengan *rule* maka paket tersebut dianggap sebagai sebuah serangan.

   b. *Behavior-based* atau *anomaly* yaitu dapat mendeteksi adanya penyusupan dengan mengamati adanya kejanggalan-kejanggalan pada sistem atau adanya penyimpangan-penyimpangan dari kondisi normal.

2. *Packet Filtering Firewall*

   Packet Filtering Firewall dapat membatasi akses koneksi berdasarkan parameter-parameter seperti *protocol,* IP asal, IP tujuan, *port* asal, *port* tujuan, dan aliran data (*chain*) sehingga dapat diatur hanya akses yang sesuai dengan *policy* saja yang dapat mengakses system. *Packet Filtering Firewall* ini bersifat statik sehingga fungsi untuk membatasi akses pun bersifat static. Untuk itulah *Packet Filtering Firewall* tidak dapat mengatasi gangguan yang bersifat dinamik sehingga harus dikombinasikan dengan IDS untuk membentuk sistem yang maksimal.

3. *Engine System*

   Engine ini bertugas untuk membaca *alert* dari IDS (dapat berupa jenis serangan dan IP *address* penyusup) untuk kemudian memerintahkan *firewall* untuk mem-*block* akses koneksi penyusup ke sistem.

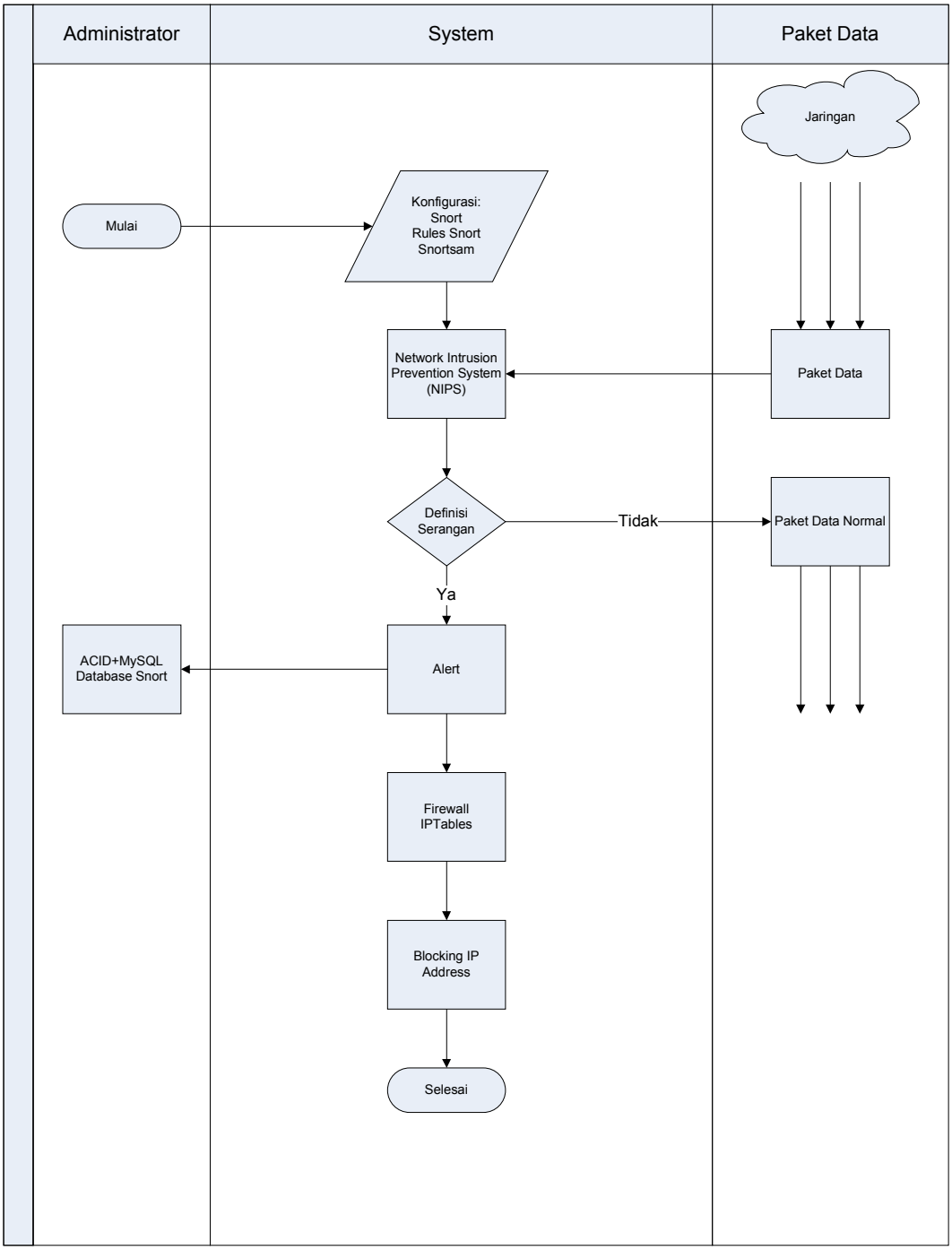
4.3 Konfigurasi Sistem Pencegahan Penyusupan Pada Jaringan (*Network Intrusion Prevention System*)

Untuk memenuhi kebutuhan fungsional sistem pencegahan penyusup pada jaringan (*Network Intrusion Prevention System*), dibutuhkan modul-modul utama untuk mendukung sistem tersebut. Modul utama berupa Snort *Engine, rule* Snort, Snortsam dan IPTables.

Target implementasi *Intrusion Prevention System* (IPS) yaitu pada sistem operasi Linux Ubuntu 10.04 (Lucid Lynx) LTS (*Long Term Support*) yang menggunakan *kernel* 2.6.28.28-*generic*.

Instalasi sistem operasi Linux Ubuntu dan konfigurasi Snort *Intrusion Prevention System* dapat dilihat pada lampiran A.

Flowchart Network Intrusion Prevention System (NIPS) dapat dilihat pada gambar berikut:



Gambar 4.3 *Flowchart Network Intrusion Prevention System*

Dari *flowchart* diatas dapat dilihat tahapan kerja *network administrator* dalam membangun *Network Intrusion Prevention System* (NIPS) meliputi:

1. Konfigurasi Snort
2. Konfigurasi *Rules* Snort
3. Konfigurasi Snortsam
4. Konfigurasi IPTables dengan Snort, *Rules* Snort dan Snortsam
5. *Log* yang dihasilkan dari pendeteksian serangan penyusupan akan disimpan pada *database* Snort yang dapat dilihat oleh *network administrator* dengan bantuan komponen pendukung berupa ACIDBASE.

BAB V

IMPLEMENTASI DAN PENGUJIAN

Tahapan selanjutnya adalah tahap implementasi dan pengujian. Pada tahap ini, hasil analisa dan perancangan sistem yang dibuat akan diimplementasikan ke dalam bentuk nyata dan kemudian akan dilakukan pengujian untuk mengetahui hasil dari analisa dan perancangan yang telah dilakukan.

5.1 Implementasi Sistem

Implementasi merupakan tahapan lanjutan setelah analisa dan konfigurasi dilakukan. Pada tahapan ini, sistem yang telah selesai, siap untuk dioperasikan dan dilakukan pengujian untuk melihat sejauh mana sistem yang dibuat dapat mencapai tujuan.

Tujuan implementasi yaitu:

1. Menyelesaikan konfigurasi sistem
2. Menguji prosedur-prosedur konfigurasi sistem
3. Mempertimbangkan bahwa sistem sesuai dengan harapan yakni menguji sistem secara keseluruhan.

Langkah-langkah yang dibutuhkan dalam pengimplementasian sistem adalah sebagai berikut:

1. Menyelesaikan konfigurasi sistem
2. Memilih komponen pendukung yang cocok dengan IDS yang digunakan
3. Mempersiapkan lingkungan implementasi
4. Menguji sistem

Terdapat tiga bagian utama yang berperan pada proses implementasi *Intrusion Prevention System.* Berikut adalah deskripsi implementasi sistem:

1. Implementasi Snort *engine*

   Memonitor *traffic* dan paket data pada jaringan.
2. Implementasi *rules* snort

   Mendeteksi dan mengelompokkan paket data yang lewat apakah merupakan sebuah serangan atau hanya paket data biasa.

3. Implementasi *output plugin*

   Snortsam merupakan *plugin* dari snort yang membuat *rule* Snort mampu memerintahkan *firewall* (IPTables) untuk mem-*block* paket data yang didefinisikan sebagai serangan penyusupan oleh Snort *engine*.

5.1.1 Lingkungan Implementasi

Lingkungan implementasi yang digunakan untuk mengkonfigurasikan Snort sebagai sistem pencegahan penyusupan pada jaringan (*Network Intrusion Prevention System*/NIPS) terdiri dari:

1. Perangkat Keras

   Perangkat keras yang digunakan memiliki spesifikasi sebagai berikut:

   a. *Processor* : Intel Core i5 2.4 GHz
   b. *Memory* : 2 GB
   c. *Harddisk* : 320 GB

2. Perangkat Lunak

   a. Sistem Operasi : Linux Ubuntu 10.04 LTS
   b. *Kernell* : 2.6.32-28-*generic*
   c. *Intrusion Detection System* : Snort 2.9.0.5
   d. *Rules* Snort : Snortrules-Snapshot-2905
   e. Komponen Pendukung : Pada Lampiran B
   f. *Output Plugin* : Snortsam
   g. *Firewall* : IPTables

3. Lain-lain

   a. Snort Sensor : eth0 – 192.168.1.1
   b. *Range* IP *address* : 192.168.1.0/24
   c. *Web Browser Monitoring* : ACIDBASE, Mozilla

5.1.2 Batasan Implementasi

Snort NIPS menguji serangan yang umum terjadi. Pengujian tipe serangan lainnya dapat dikondisikan menurut *rules* snort.

5.1.3 Hasil Implementasi

Hasil implementasi adalah pencegahan serangan yang terjadi pada sistem pencegahan penyusupan pada jaringan (*Network Intrusion Prevention System*) dan menampilkan *output* monitoring pada *database*.

5.2 Pengujian *Network Intrusion Prevention System* (NIPS)

Untuk menguji sistem pencegahan penyusupan, dilakukan dengan cara melancarkan paket serangan ke sistem yang dilindungi oleh sistem pencegahan penyusupan (*Network Intrusion Prevention System*).

5.2.1 Lingkungan Pengujian

Lingkungan pengujian yang digunakan untuk menguji sistem pencegahan penyusupan pada jaringan (*Network Intrusion Prevention System*/NIPS) terdiri dari:

1. Perangkat Keras
   a. *Processor* : Intel Core 2 Duo 1.73 Ghz
   b. *Memory* : 2 GB
   c. *Harddisk* : 120 GB
2. Perangkat Lunak
   a. Sistem Operasi : Linux Backtrack 4 R2
   b. Aplikasi Pengujian : Backtrack *tools*
   c. IP *address* : 192.168.1.2

5.2.2 Identifikasi dan Rencana Pengujian Snort *Network Intrusion Prevention System* (NIPS)

Identifikasi dan rencana pengujian dapat dilihat pada table 5.1

Tabel 5.1 Identifikasi dan Rencana Pengujian Snort *Network Intrusion Prevention System* (NIPS)

Kelas Uji	Butir Uji	Tingkat Pengujian	Jenis Pengujian	jadwal
Percobaan serangan penyusupan pada Snort NIPS	Normal	Pengujian *Network Intrusion Prevention System*	*Blackbox*	30 Juni 2011

5.2.3 Pengujian *Host Scanning*

Pada pengujian ini dilakukan percobaan pengamatan terhadap *host* yang ada pada jaringan. Berikut contoh pengujian serangan yang dilakukan menggunakan aplikasi Angry IPScan.

1. Kondisi Snort *Network Intrusion Prevention System* tidak aktif

Gambar 5.1 Percobaan *host scanning* sukses

Dari pengamatan diatas terlihat bahwa pengujian *host scanning* pada *range* IP *address* 192.168.1.1 ke 192.168.1.10 berhasil ketika snort tidak dalam keadaan aktif. Dengan melakukan *host scanning*, penyerang bisa mengetahui *host*-host

mana saja dalam keadaan aktif. Pada percobaan diatas, IP *address server* NIPS yaitu: 192.168.1.1 dan IP address penyerang: 192.168.1.2.

2. Kondisi Snort *Network Intrusion Prevention System* aktif

Gambar 5.2 Percobaan *host scanning* gagal

Dari pengamatan diatas terlihat bahwa percobaan *host scanning* gagal dengan tidak dijumpai IP *address server*. Yang terlihat hanya IP *address* dari penyerang.

Tabel 5.2 Butir Uji Modul Pengujian *host scanning* pada *Server* Snort NIPS

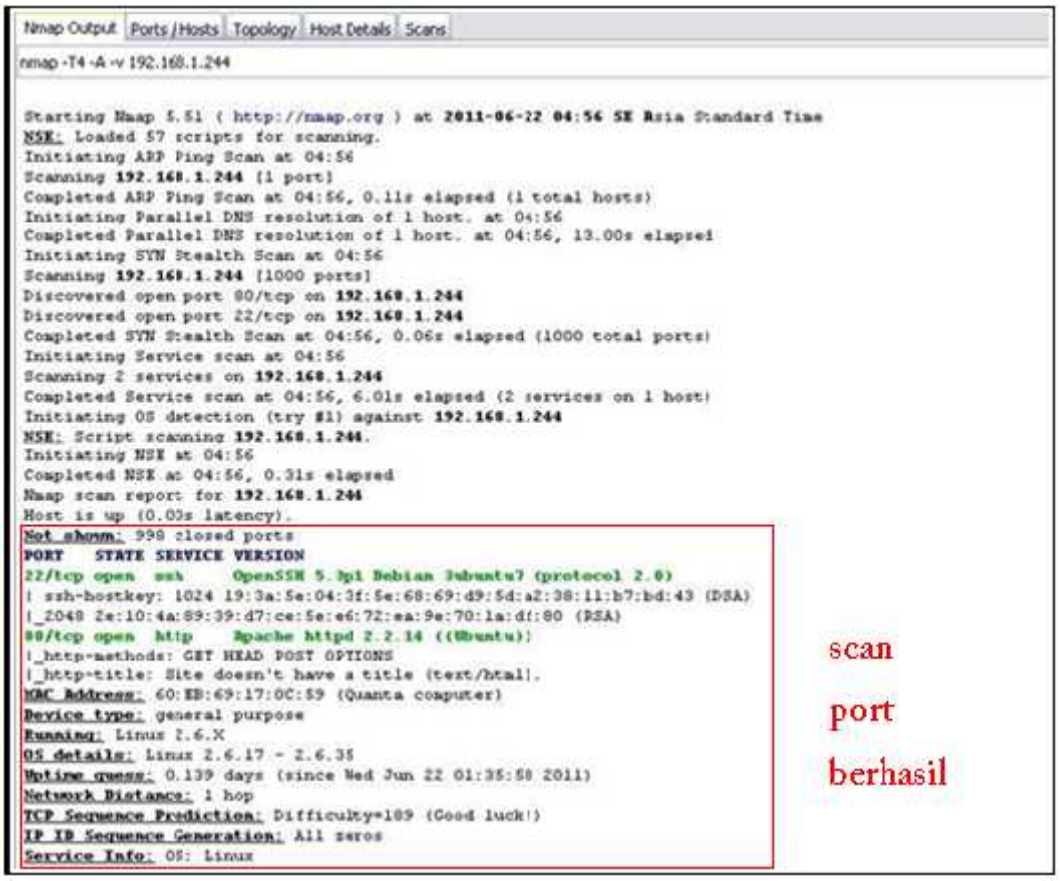
Deskripsi dan Prekondisi	Tools dan IP *address* penyusup	Prosedur Pengujian	Keluaran yang diharapkan	Hasil dan Kesimpulan
Snort *Network Intrusion Prevention System* (NIPS) dalam keadaan aktif	Angry IPScan 192.168.1.2	Penyusup melakukan *scanning host* yang aktif	Tampil IP *Address* penyusup dalam bentuk *alert* dan sistem melakukan *blocking* IP *address* dan serangan	Tampil IP *address* penyusup dalam bentuk *alert* pada terminal dan acidbase Kesimpulan pengujian : Berhasil

5.2.4 Pengujian *Port Scanning*

Pada pengujian ini dilakukan percobaan pengamatan *port-port* yang terbuka. Berikut contoh pengujian serangan yang dilakukan menggunakan aplikasi zenmap.

1. Pengamatan di komputer penyerang

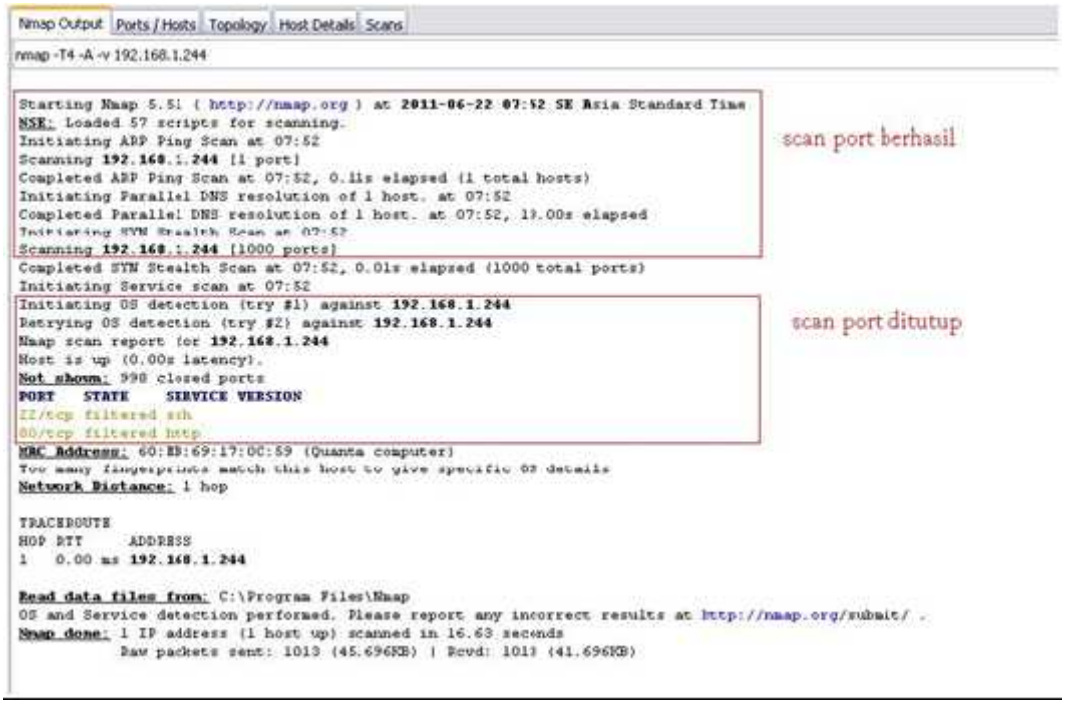
a. Kondisi Snort *Network Intrusion Prevention System* tidak aktif



Gambar 5.3 Percobaan *port scanning* sukses

Dari pengamatan diatas terlihat bahwa pengujian *port scanning* ke server snort NIPS berhasil ketika snort tidak dalam keadaan aktif. Dengan melakukan *port scanning*, penyerang bisa mengetahui *port-port* mana saja yang terbuka dan dapat diakses. Dari gambar diatas terlihat bahwa *port* 22 (ssh) dan *port* 80 (HTTP) terbuka dan dapat diakses.

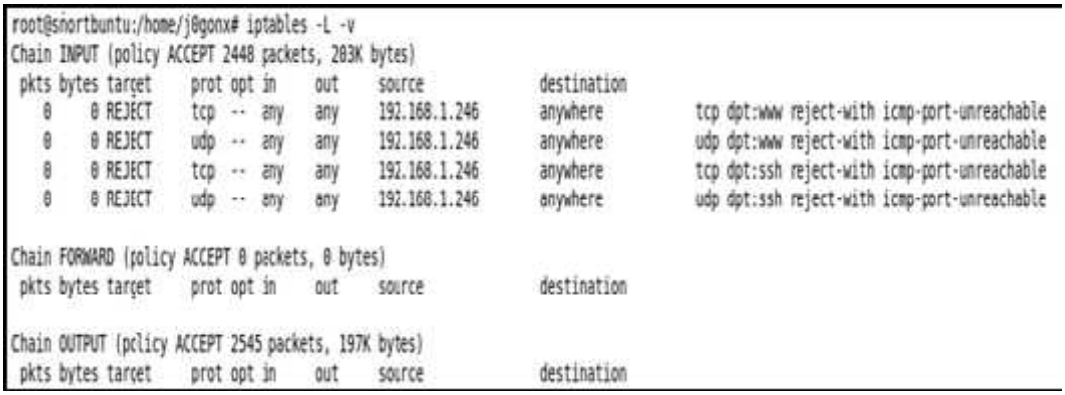
b. Kondisi Snort *Network Intrusion Prevention System* aktif



Gambar 5.4 Percobaan *port scanning* gagal

Dari gambar di atas terlihat bahwa snort NIPS telah bekerja dengan baik, hal ini ditunjukkan dengan tertutupnya *port* 22 (ssh) dan *port* 80 (HTTP) ditandai dengan *filtered*.

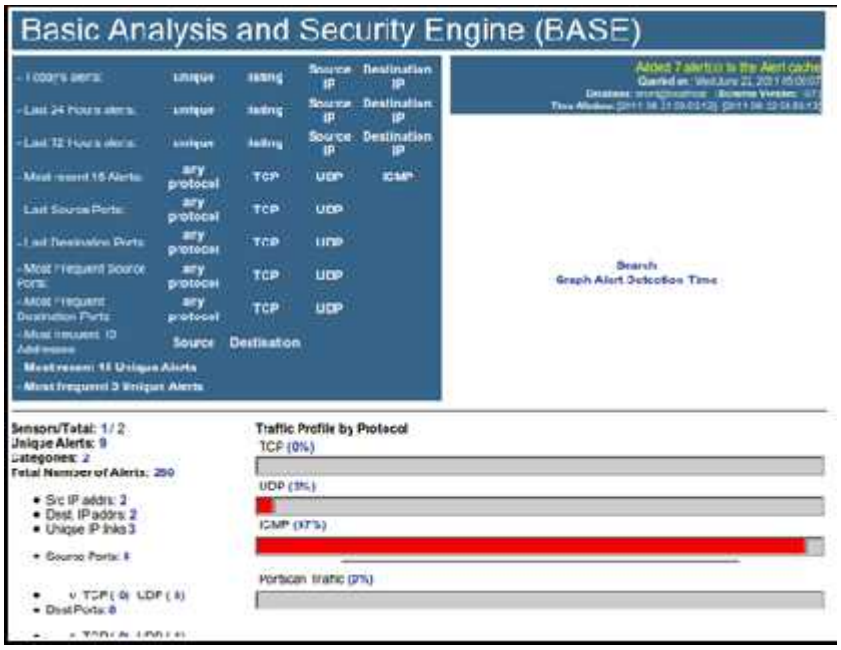
c. Pengamatan di *server Network Intrusion Prevention System*



Gambar 5.5 Pengamatan pengamanan

Dari pengamatan yang dilakukan di kedua sisi, baik sisi penyerang dan sisi sistem pencegahan penyusupan, dapat disimpulkan bahwa fungsional sistem ini telah berjalan seperti yang dirancang. Serangan-serangan jenis lain juga akan di *block* oleh sistem tergantung kelengkapan *rule* snort.

d. Pengamatan *Database* Snort Melalui Acidbase



Gambar 5.6 *Alert* yang disimpan ke dalam *database*

Tabel 5.3 Butir Uji Modul Pengujian *port scanning* pada *Server* Snort NIPS

Deskripsi dan Prekondisi	Tools dan IP *address* penyusup	Prosedur Pengujian	Keluaran yang diharapkan	Hasil dan Kesimpulan
Snort *Network Intrusion Prevention System* (NIPS) dalam keadaan aktif	Zenmap 192.168.1.2	Penyusup melakukan *scanning port* yang aktif	Tampil IP *Address* penyusup dalam bentuk *alert* dan sistem melakukan *blocking* IP *address* dan serangan	Tampil IP *address* penyusup dalam bentuk *alert* pada terminal dan acidbase

5.2.5 Pengujian Akses *Localhost*

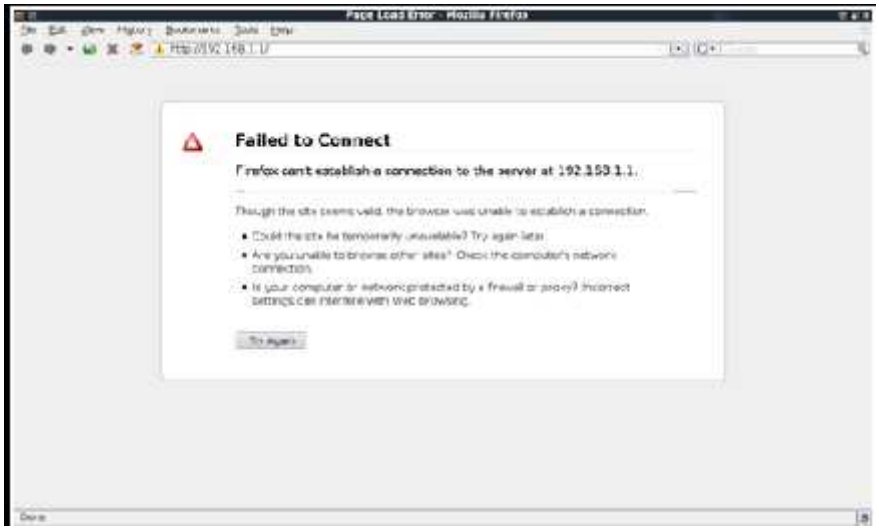
Pada pengujian ini, penyerang akan mengakses *localhost* dari Snort NIPS.

1. Kondisi Snort *Network Intrusion Prevention System* tidak aktif.

Gambar 5.7 Akses *localhost* berhasil

Dari pengamatan diatas dapat dilihat bahwa akses ke *localhost* ke *server* NIPS berhasil ketika Snort dalam keadaan tidak aktif.

2. Kondisi Snort *Network Intrusion Prevention System* aktif



Gambar 5.8 Akses *localhost* gagal

Dari pengamatan diatas dapat dilihat bahwa akses ke *localhost* ke *server* NIPS gagal ketika Snort dalam keadaan aktif.

Tabel 5.4 Butir Uji Modul Pengujian *HTTP Attacking* pada *Server* Snort NIPS

Deskripsi dan Prekondisi	Tools dan IP *address* penyusup	Prosedur Pengujian	Keluaran yang diharapkan	Hasil dan Kesimpulan
Snort *Network Intrusion Prevention System* (NIPS) dalam keadaan aktif	Mozilla Web Browser 192.168.1.2	Penyusup melakukan *HTTP Attacking* pada server NIPS	Tampil IP *Address* penyusup dalam bentuk *alert* dan sistem melakukan *blocking* IP *address* dan serangan	Tampil IP *address* penyusup dalam bentuk *alert* pada terminal dan acidbase Kesimpulan pengujian : Berhasil

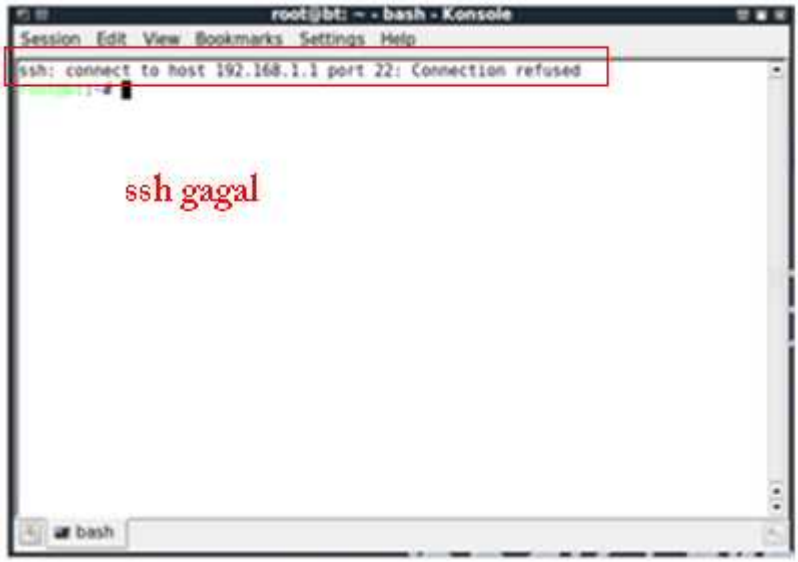
5.2.6 Pengujian Akses SSH

Pada pengujian ini, penyerang akan mengakses Snort NIPS melalui SSH

1. Kondisi Snort *Network Intrusion Prevention System* tidak aktif

Gambar 5.9 Akses SSH berhasil

2. Kondisi Snort *Network Intrusion Prevention System* aktif



Gambar 5.10 Akses SSH gagal

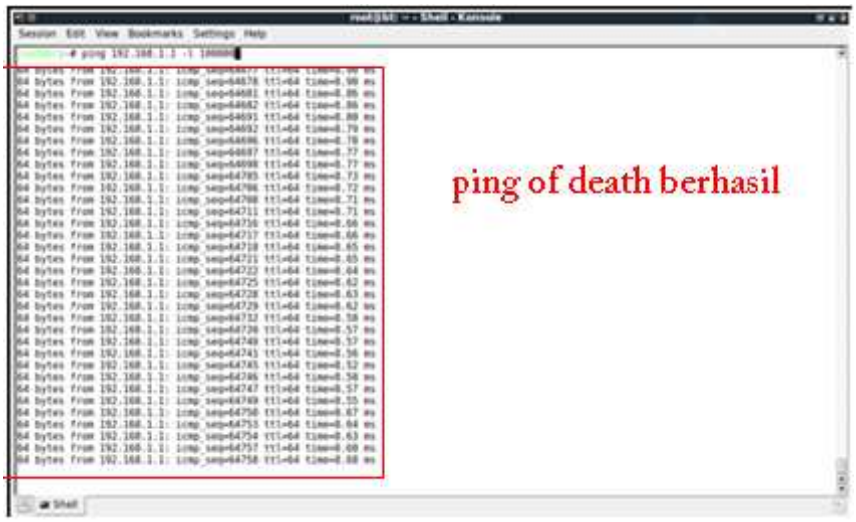
Tabel 5.5 Butir Uji Modul Pengujian *SSH Attacking* pada *Server* Snort NIPS

Deskripsi dan Prekondisi	Tools dan IP *address* penyusup	Prosedur Pengujian	Keluaran yang diharapkan	Hasil dan Kesimpulan
Snort *Network Intrusion Prevention System* (NIPS) dalam keadaan aktif	SSH 192.168.1.2	Penyusup melakukan *login SSH* pada server NIPS	Tampil IP *Address* penyusup dalam bentuk *alert* dan sistem melakukan *blocking* IP *address* dan serangan	Tampil IP *address* penyusup dalam bentuk *alert* pada terminal dan acidbase Kesimpulan pengujian : Berhasil

5.2.7 Pengujian *Ping of Death*

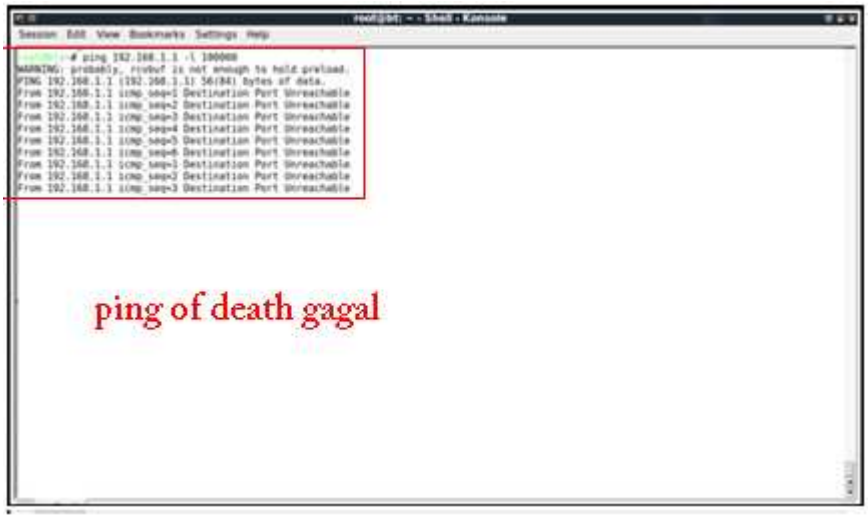
Pada pengujian ini, penyerang akan melakukan *Denial of Service* (DoS) berupa *Ping of Death*, yaitu dengan mengirimkan paket ICMP dalam jumlah besar ke *server* NIPS.

1. Kondisi Snort *Network Intrusion Prevention System* tidak aktif



Gambar 5.11 *Ping of Death* berhasil

2. Kondisi Snort *Network Intrusion Prevention System* aktif



Gambar 5.12 *Ping of Death* gagal

Tabel 5.6 Butir Uji Modul Pengujian *Ping of Death* pada *Server* Snort NIPS

Deskripsi dan Prekondisi	Tools dan IP *address* penyusup	Prosedur Pengujian	Keluaran yang diharapkan	Hasil dan Kesimpulan
Snort *Network Intrusion Prevention System* (NIPS) dalam keadaan aktif	Terminal 192.168.1.2	Penyusup melakukan *ping of death* pada server NIPS	Tampil IP *Address* penyusup dalam bentuk *alert* dan sistem melakukan *blocking* IP *address* dan serangan	Tampil IP *address* penyusup dalam bentuk *alert* pada terminal dan acidbase Kesimpulan pengujian : Berhasil

5.2.8 Analisa Hasil Pengujian

Analisa hasil pengujian Snort *Network Intrusion Prevention System* (NIPS) dapat dilihat pada table 5.7.

Tabel 5.7 Hasil Pengujian

Implementasi	Kelas Uji	Hasil	Deskripsi
Snort *Network Intrusion Prevention System* (NIPS)	Percobaan serangan penyusupan	Sesuai	Menghasilkan analisa yang sesuai dengan metode yang digunakan dan konfigurasi yang dilakukan

5.3 Kesimpulan Pengujian

Cara kerja sistem pencegahan penyusupan pada jaringan (*Network Intrusion Prevention* System) ini bekerja dengan bantuan Snortsam dan IPTables sebagai *firewall*. Snortsam adalah sebuah *plugin* Snort yang membuat *rule* Snort mampu memerintahkan IPTables untuk mem-*block* suatu paket data. Ketika Snort memonitoring paket data, *rule* Snort mengidentifikasikan paket data tersebut apakah sebagai sebuah serangan penyusupan atau hanya berupa paket data biasa, tergantung dari *rule* Snort itu sendiri. Jika paket data digolongkan sebagai serangan penyusupan oleh *rule* Snort, maka *rule* Snort akan memerintahkan IPTables mem-*block* serangan penyusupan tersebut.

Snort *Network Intrusion Prevention System* (NIPS) sangat cocok digunakan untuk me-*monitoring* jaringan secara *real-time*, dapat mendeteksi adanya serangan penyusupan yang terjadi pada jaringan dan mencegah secara otomatis serangan penyusupan tersebut. Snort NIPS juga dapat dikonfigurasikan dengan modul tambahan berupa Acidbase untuk mendukung dokumentasi atau laporan hasil *monitoring* jaringan yang berguna bagi *network administrator* dalam menganalisa paket data yang ada pada jaringan.

BAB VI

PENUTUP

Tahap akhir dari penulisan laporan ini adalah penutup. Tahap ini berisi kesimpulan dan saran yang didapat ketika telah selesai melaksanakan konfigurasi dan pengujian sistem secara keseluruhan.

6.1 Kesimpulan

Berdasarkan pembahasan pada bab-bab sebelumnya dan merujuk pada tahap konfigurasi dan implementasi, dapat diambil beberapa kesimpulan, yakni:

1. Snort *Network Intrusion Prevention System* (NIPS) sangat cocok dikonfigurasikan pada sistem operasi Linux Ubuntu karena kemudahan dalam penggunaan dan kecocokan komponen utama dan komponen pendukung untuk mengkonfigurasikan Snort NIPS pada sistem operasi Linux Ubuntu.
2. Snort NIPS bekerja dengan cara membangun sebuah Snort *engine* yang memonitor paket data dan mencocokkannya pada *rule* Snort. Jika paket data diidentifikasi sebagai serangan, *rule* akan memerintahkan *firewall* (IPTables) untuk mem-*block* serangan tersebut.
3. Suatu serangan dapat terdeteksi atau tidak oleh Snort *engine* tergantung pada pola serangan yang ada pada *rule database* snort. Jika pola serangan tidak terdefinisi pada *rule database*, maka serangan akan dianggap sebagai paket data biasa.
4. Snort *Network Intrusion Prevention System* (NIPS) mampu mendeteksi dan melakukan pencegahan terhadap serangan penyusupan berupa *host scanning* menggunakan *tool Angry IPScan*, *port scanning* menggunakan *tool Zenmap*, *http attacking* menggunakan *tool Mozilla*, *ssh attacking* dan *ping of death* menggunakan *terminal* pada sistem operasi Linux Backtrack.

6.2 Saran

Saran untuk pengembangan Snort selanjutnya adalah:

1. Mengkonfigurasikan Snort secara otomatis dengan tampilan *Graphical User Interface* (GUI) agar memudahkan *network administrator* dalam mengelola dan meng-*update* Snort dan *Rules* Snort.
2. Snort juga dapat digunakan sebagai sistem pencegah penyusupan dengan konfigurasi sms *server* agar pemberitahuan tentang penyusupan pada jaringan dapat segera diketahui oleh *network administrator* melalui pesan sms.

DAFTAR PUSTAKA

Alder, Raven. *Snort 2.1 Intrusion Detection*, *Second Edition*. Rockland, MA 02370: Syngress Publishing, Inc. 2004

————— . *Snort IDS and IPS Toolkit*. Burlington, MA 01803: Syngress Publishing, Inc. 2007

Babbin, Jacob. *Snort Cookbook*. 1005 Gravenstein Highway North, Sebastopol, CA: O'Reilly Media, Inc. 2005

Fahrial, Jaka, *Teknik Konfigurasi LAN di Windows*, Ilmu Komputer, www.ilmukomputer.com/paper/08/2004/jaka-fahrial-teknik-kofigurasi-lan-diwindows.pdf, agustus 2004, diakses pada tanggal 14 Maret 2011

Gullett, David, *Snort 2.9.0.5 and Snort Report 1.3.1 on Ubuntu 10.04 LTS Installation Guide*. United States: Symmetrix Technologies. 2011

Rafiudin, Rahmat, *Panduan Membangun Jaringan Komputer untuk Pemula*. Jakarta: PT. Elex Media Komputindo. 2003

Rehman, Rafeeq Ur. *Intrusion Detection System with Snort: Advance IDS Techniques with Snort, Apache, Mysql, PHP, ACID*. Upper Saddle River, New Jersey 07458: Pearson Education, Inc. 2003

The Snort Project. *Snort User Manual 2.9.0*. United States: Sourcefire, Inc. 2011

Von Hagen, William. *Ubuntu Linux Bible*. Indiana: Wiley Publishing, Inc. 2007

Waskita, Adi. Hiswendari, Lely. *Local Area Network*: *Basic*.Jakarta. 2004